EFFICIENT AUDIO SIGNAL PROCESSING FOR EMBEDDED SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Leung Kin Chiu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2012

EFFICIENT AUDIO SIGNAL PROCESSING FOR EMBEDDED SYSTEMS

Approved by:

David V. Anderson, Advisor
Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Jennifer O. Hasler
Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Aaron D. Lanterman
Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

William D. Hunt
Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Bradley A. Minch
Professor, Electrical and Computer Engineering
Franklin W. Olin College of Engineering

Date Approved: May 16, 2012

To my family

ACKNOWLEDGEMENTS

I am grateful for having the opportunity to come to the United States when I was a teenager. I appreciate the educational system in this country, which not only values creativity and promotes personal growth, but also offers plenty of "second chances" if one does not excel the first time. I also appreciate the American ideal of achieving social upward mobility not through *guanxi* but through hard work.

I would like to express my immense gratitude to my advisor, David Anderson, for his efforts to help me succeed. He allowed me the intellectual freedom to develop my research topics and guided me when I was lost. By being a great role model, he has influenced me in academia and beyond. I am grateful to Jennifer Hasler for introducing me to the world of programmable analog systems and neuromorphic engineering. She has challenged me to set an even higher bar for my research goals. I would like to thank Bradley Minch for carefully reading my thesis and giving me detailed feedback to improve it. I would also like to thank Aaron Lanterman and William Hunt for serving on my defense committee and providing helpful suggestions on my thesis and presentation style.

I want to thank the many professors who have impacted my life during my undergraduate and graduate studies. In particular, I would like to express my thanks to Stephen Arnold, Rajit Manohar, Sheila Hemami, and Justin Romberg for being inspiring teachers.

I am fortunate to have met a number of mentors without whom I would not have made it this far. Among them, my special thanks go to Kent Foo, for encouraging me to attend college, and John F. Hamilton, for persuading me to pursue a Ph.D.

Members of the Efficient Signal Processing (ESP) Lab have been excellent company for the past six years, and I have enjoyed the research and social atmosphere of the lab. In particular, I would like to thank Brian Gestner, Devangi Parikh, Jorge Marin, Nathan Parrish, Ailar Javadi, Ryan Curtin, Abbie Kressner, Spence Whitehead, Tira Duangudom, Chu Meh Chu, Rich Hammett, Woosuk Choi, Bo Marr, Faik Baskaya, Walter Huang, Jungwon Lee, Hawjing Lo, Nikolaos Vasiloglou, Shyam Subramanian, and Taehyung Lee.

My work on programmable analog systems would not have been possible without the help from members of the Integrated Computational Electronics (ICE) Lab. I especially enjoyed the insightful discussions with Shubha Ramakrishnan, Craig Schlottmann, Stephen Brink, Steve Nease, Samuel Shapero, Arindam Basu, Brian Degnan, Scott Koziol, Jordan Gray, Ryan Robucci, Richie Wunderlich, Farhan Adil, and Christal Gordon.

I would like to acknowledge Keith Jemo, Brandon Miller, Scott Kammlade, and Chris Broussar, my teammates in the Technological Innovation: Generating Economic Results (TI:GER) Program, for investigating the commercialization potential of my research. I also thank Marie Thursby, Margi Berbari, and Anne Rector for teaching me the business and legal aspects of technological innovation.

Over the years, I have interacted with many colleagues and fellow students who tended to work late into the night in Tech Square Research Building (TSRB). I thank Debesh Bhatta, Jenna Foo, Logan Sorenson, Vikram Appia, Xiaolin Liang, Mauricio Pardo, Milap Dalal, Xueli Xiao, Hoseon Lee, and Hakan Töreyin for their camaraderie.

It has been a great pleasure to coordinate the weekly seminars for the Center for Signal and Image Processing (CSIP). I also enjoyed the interactions with many CSIP students, especially Salman Asif, Umair Altaf, Chris Turnes, Aurèle Balavoine, William Mantzel, Aditya Joshi, Adam Charles, Sourabh Khire, Hrishikesh Rao, and Kaustubh Kalgaonkar.

I want to express my thanks to Janet Arnold, Nydia Akins, and Stacie Speights for their great administrative supports in TSRB and CSIP.

Other groups of the Georgia Tech community have made my stay here more enjoyable. Members of the Georgia Tech Ballroom Dance Club and Georgia Tech Salsa Club have provided me the much-needed healthy distractions, and I am particularly grateful to Sarah Watson, Yichen Fang, Didi Eze, and Gena Tang. I have been fortunate to join a Saturday Cooking Night group with fellow graduate students Jeff Wong, Samson Lai, Adrian Lam, Elaine Tang, Linda Hui, Samantha Lo, and others. I truly appreciate their companionship.

I would like to thank my parents and sister for their encouragement through the long process of completing a Ph.D. Last but not least, this thesis would not have been possible without the love and support of Wai Man Chan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

As mobile platforms continue to pack on more computational power, electronics manufacturers start to differentiate their products by enhancing the audio features. However, consumers also demand smaller devices that could operate for longer time, hence imposing design constraints. In this research, we investigate two design strategies that would allow us to efficiently process audio signals on embedded systems such as mobile phones and portable electronics. In the first strategy, we exploit properties of the human auditory system to process audio signals. We designed a sound enhancement algorithm to make piezoelectric loudspeakers sound "richer" and "fuller." Piezoelectric speakers have a small form factor but exhibit poor response in the low-frequency region. In the algorithm, we combine psychoacoustic bass extension and dynamic range compression to improve the perceived bass coming out from the tiny speakers. We also developed an audio energy reduction algorithm for loudspeaker power management. The perceptually transparent algorithm extends the battery life of mobile devices and prevents thermal damage in speakers. This method is similar to audio compression algorithms, which encode audio signals in such a ways that the compression artifacts are not easily perceivable. Instead of reducing the storage space, however, we suppress the audio contents that are below the hearing threshold, therefore reducing the signal energy. In the second strategy, we use low-power analog circuits to process the signal before digitizing it. We designed an analog front-end for sound detection and implemented it on a field programmable analog array (FPAA). The system is an example of an analog-to-information converter. The sound classifier front-end can be used in a wide range of applications because programmable floating-gate transistors are employed to store classifier weights. Moreover, we incorporated a feature selection algorithm to simplify the analog front-end. A machine learning algorithm AdaBoost is used to select the most relevant features for a particular sound detection application. In this classifier architecture, we combine simple "base" analog classifiers to form a strong one. We also designed the circuits to implement the AdaBoost-based analog classifier.

CHAPTER I

INTRODUCTION

As electronic devices become ever thinner, smaller, and lighter, consumers demand ever higher performance and more functionalities from these devices. In addition, power consumption is a design constraint in portable electronics. In this work, we explore two strategies to design efficient systems under these conflicting design goals.

In the first strategy, we take advantage of the fact that, in signal processing applications, many of the signals are processed for human consumption. Therefore, we design signal processing systems to exploit the way humans perceive sound and image, allocating resources to process information that we can actually hear and see. Specifically, we focus on audio signal processing and employ psychoacoustic techniques in processing audio signals. Two systems based on real-world applications are presented. The first system enhances low frequency sound in small piezoelectric loudspeakers. Its design is based on the psychoacoustic principles of "bass extension" and "perceived loudness." The second system reduces audio power in a perceptually transparent fashion. It is applied in loudspeaker power anagement and is based on the psychoacoustic principle of "masking."

In the second strategy, we design a signal processing system on a low-power analog platform. There have been many arguments for the benefits of exploiting analog techniques to design low-power systems. For example, Furth and Andreou compared analog and digital circuits from an information theoretic perspective and concluded that for low values of power dissipation, analog circuits have a superior performance in terms of channel capacity [34]. Sarpeshkar compared analog and digital circuits from a noise perspective and analyzed the costs of achieving high precision in these two platforms. He concluded that for a given chip area, analog circuits consume less power at low signal-to-noise ratio (SNR) at the output [89]. More recently, researchers analyzed the trade-off of "analog versus digital" from a system design perspective. The real world is mostly analog—we see analog images and hear

analog sounds. However, most of the computations are done in the digital domain. Hasler and Anderson proposed an alternative design approach in which part of the signal processing operations are performed in the analog domain, and only the useful information is converted into the digital domain [43]. This paradigm is appropriately called *cooperative analog/digital signal processing* (CADSP). It is particularly useful if low-precision computation is required in the front-end because it can be done very efficiently in the analog domain. We present the analog front-end of a sound detector to exemplify this design strategy.

This thesis is organized as follows:

- **Part 1: Psychoacoustic-based audio signal processing**

  In Chapter 2, an audio output enhancement algorithm for piezoelectric loudspeakers is presented. Piezoelectric loudspeakers have a small form factor and are suitable for electronic devices that have a thin profile. However, they have poor frequency response, especially at low frequencies. In this chapter, we introduce a method that combines psychoacoustic bass extension and sub-band dynamic range compression to make the audio sound "richer" in piezoelectric speakers. Subjective test results are also presented.

  In Chapter 3, a perceptually transparent audio power reduction algorithm for loudspeaker power management is described. Power management in loudspeakers is used to extend the battery life of portable electronics and to prevent thermal damage to the speakers. Traditionally, speaker power management has been implemented as an adaptive gain control, which limits the power applied to the loudspeaker uniformly. In this chapter, we present a perception-based algorithm that suppresses inaudible contents from the audio before sending it to the amplifier and loudspeaker. The power reduction algorithm attenuates the signal energy in the digital domain. Subjective test results show that the energy reduction is largely unnoticeable in small loudspeakers.

- **Part 2: CADSP-inspired sound classifier design**

  In Chapter 4, we review the two components of a sound classifier, *audio feature extraction* and *classification*. In the feature extraction stage, the input audio signal is

represented in distinguishable forms. The set of extracted features is known as the feature vector. In the classification stage, the feature vector is analyzed, and the input signal is classified into different categories.

In Chapter 5, the analog implementations of sound classifiers are discussed. Low-power analog classifiers are typically sophisticated analog very-large-scale integration (VLSI) systems that are implementations of powerful pattern classification algorithms. In this chapter, we review the design techniques reported in literature.

In Chapter 6, we present the design of an analog front-end for sound detection on a programmable analog platform. In this architecture, we employ sub-band energy estimation and zero-crossing rate as audio features; for classification, we use a simple multiple-input perceptron. We show that the system is suitable for analog design by demonstrating its implementation on a reconfigurable analog platform—the field-programmable analog array (FPAA), which is fabricated in a 0.35-µm CMOS technology. This design is suitable to be placed before the analog-to-digital converter (ADC) in an intelligent audio sensor. We highlight a security application of the system in detecting glass-break sound.

In Chapter 7, the design of analog audio classifiers with AdaBoost-based feature selection is discussed. The design of analog classifiers constitutes a trade-off between performance and complexity, and designers have historically adopted more complex architectures to lower the error rate of a classification task. An alternative design paradigm is presented in this chapter: We design the front-end of a sound classification system with simple "base" classifiers. We then enhance the overall performance with the aid of the AdaBoost algorithm, which selects the most appropriate "base" classifiers and combines them with different weights. We describe the general architecture and the algorithm to select features and present a design example with simulation results in a TSMC-compatible 0.35-µm technology.

In Chapter 8, we discuss two designs of the AdaBoost-based analog classifier. Both designs can be used as the classification stage of the sound detector front-end presented

in Chapter 7. In the first design, the circuit is an extension of the multiple-input perceptron presented in Chapter 6. In the second design, we simplify the circuit by implementing a shifted step function for the single-input perceptron and controlling the weighted current sources in the second layer with binary inputs. The first design is verified in SPICE, and the second design is verified in hardware using the reconfigurable analog signal processor (RASP) 2.8a.

- Concluding remarks are made in Chapter 9. We present a summary of contributions in this research and suggest future research based on the current work.

CHAPTER II

SOUND ENHANCEMENT FOR PIEZOELECTRIC LOUDSPEAKERS

The demand for thin speakers has increased in recent years because of the continued effort to miniaturize electronic devices. Piezoelectric loudspeakers have become a popular choice in such products as flat-panel televisions and portable music players because they are light, thin, and more power-efficient than standard dynamic speakers [49]. However, piezoelectric speakers have poor frequency response, especially at low frequencies. As a result, they are perceived to produce "tiny" sound.

There have been research efforts to improve the response of piezoelectric speakers. For example, a novel diaphragm is proposed in [74] to amend the low-frequency performance, but this approach may increase the manufacturing cost and product size. In this chapter, we investigate a solution based on signal processing. In particular, we use a combination of two techniques—psychoacoustic bass extension and dynamic range compression—to make the piezoelectric speakers sound "richer."

The bass extension algorithm creates "virtual pitches" in the low-frequency region, hence improving the bass response. The dynamic range compression algorithm increases the perceived loudness of the audio signal by reducing the peak-to-RMS ratio of the signal. The system employs a sub-band approach; input signals are broken down into critical bands before the processing. This is done to prevent undesirable artifacts associated with the two algorithms. The proposed overall system architecture is shown in Figure 1.

2.1 Piezoelectric Loudspeakers

Piezoelectric loudspeakers are a type of piezoelectric transducer, which converts an electric voltage into mechanical movements. The sound source in these components is a piezoelectric diaphragm. The diaphragm has electrodes on both sides. When a DC voltage is applied across the electrodes, the diaphragm expands; when the DC voltage is applied in the opposite polarity, it shrinks. The diaphragm is usually attached to a metal plate, which

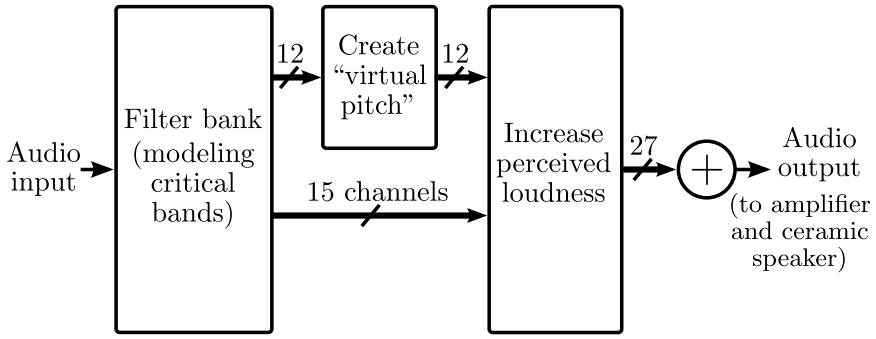


Figure 1: Overall architecture of the proposed sound enhancement system for piezoelectric speakers. A filter bank divides the input signal into 27 constant-Q critical bands. The lower 12 bands are used for psychoacoustic bass extension. All of the sub-bands undergo dynamic range compression.

does not expand or shrink. Instead, it translates the motion into bending in the directions perpendicular to the surface. Therefore, when an AC voltage is applied to the electrodes, the diaphragm-metal fixture bends repeatedly and radiates audible sound waves.

Compared to conventional magnetically actuated transducers such as the moving-coil speaker, piezoelectric speakers are designed to be thin (about 1 mm) and light (about 2 g). As piezoelectric speakers have become increasingly popular in recent years, there have been research efforts to improve their design [74]. Mono and stereo amplifiers driving the piezoelectric speakers are also commercially available.

Piezoelectric speakers have several drawbacks; the most easily perceivable one is their poor sound radiation performance in the low-frequency region. This is often attributed to the diaphragms being too stiff to vibrate with large amplitude. The piezoelectric diaphragm can be modeled by a single-resonance system (as a direct-radiator speaker); the system is in the *stiffness-controlled region* below the diaphragm resonance frequency, and the *mass-controlled region* above [49]. In the mass-controlled region, the output sound pressure is largely independent of the frequency. However, in the stiffness-controlled region, the sound pressure level is proportional to the square of the frequency because the diaphragm displacement is inversely proportional to the signal frequency. The characterization is depicted in Figure 2.

To maintain an equal loudness level for all frequencies, the resonance frequency should be

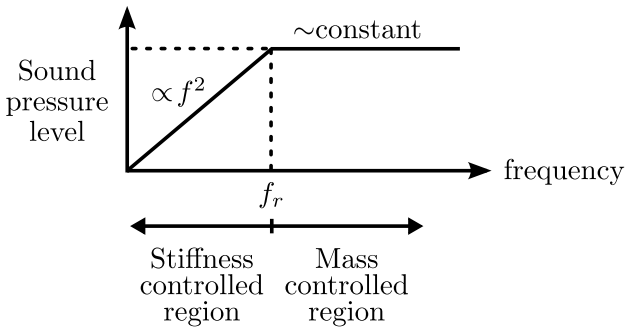


Figure 2: Frequency response of an *ideal* piezoelectric speaker, modeled as a single-resonance system [49]. Below the resonance frequency f_r, the sound pressure level of the system is proportional to f^2. Above f_r, the sound pressure level is approximately independent of the frequency.

kept as low as possible, but this is a challenge in piezoelectric transducer design. Commercially available ceramic piezoelectric speakers have a diaphragm resonance around 1 kHz. Audio signals below this frequency are inaudible or sound significantly subdued. We address signal processing methods to combat the degradation in the next two sections.

2.2 Psychoacoustic Bass Extension

Psychoacoustic bass extension algorithms take advantage of the human auditory system by "tricking" our ears into believing that bass exists in the audio signal even though the system is physically unable to produce the bass. Many of these algorithms exploit the *missing fundamental effect*, which is also known as the *virtual pitch theory*. The theory states that the human auditory system creates a pitch at frequency f_0 if presented with a harmonic series that has a common fundamental frequency. This phenomenon is illustrated in Figure 3.

According to the virtual pitch theory, to produce the *perception* of a pure tone with frequency at f_0 from a loudspeaker that has a low frequency cutoff at f_c, where $f_0 < f_c$, one can present a series of harmonics $f_1 = 2 \cdot f_0$, $f_2 = 3 \cdot f_0$, $f_3 = 4 \cdot f_0$, etc., where $f_1 > f_c$. In this case, an average human observer perceives the fundamental pitch at f_0. A slight change in the timbre of the tone can often be detected [71]. Similarly, we can generate a series of higher harmonics that are above the cutoff frequency of the loudspeaker to induce the perception of low frequency sounds in a loudspeaker that cannot physically reproduce deep

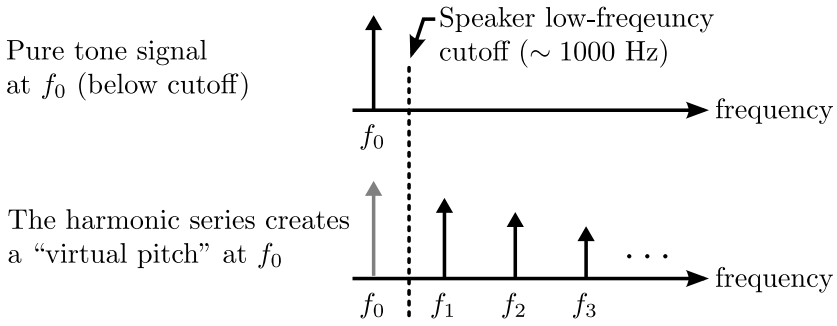


Figure 3: The "missing fundamental" phenomenon. To an average human observer, the higher harmonics of a pure tone is perceived as the pure tone itself, with a slight change in timbre. This effect is used to create an illusion of an audio signal that is below the cutoff frequency of the speaker.

bass. As a result, an average human observer would perceive the original low-frequency sounds.

The bass bandwidth extension algorithms have been applied to many systems, such as small dynamic speakers (to enhance sound quality) and sound equipments at nightclubs in serene neighborhoods (to eliminate vibration associated with deep bass) [58]. The processing usually includes extracting the low-frequency content in the original signal and generating higher harmonics with nonlinearities. The algorithms work well with the "typical" target sound systems whose low-frequency cutoff is at about 200 Hz. The piezoelectric speakers, however, have a higher cutoff frequency around 1 kHz. For signals below a few hundred Hertz (such as the music of some bassoons and trombones), some of the higher harmonics are still below the cutoff the speakers. In addition, blindly increasing the sound level of the higher harmonics causes unacceptable distortion. Therefore, psychoacoustic bass extension alone cannot fully compensate for the poor low-frequency response of the piezoelectric speakers.

2.3 Increasing Perceived Loudness

A simple way to increase the perceived loudness of an audio signal is simply to amplify the signal prior to sending it to the loudspeaker. However, this is often not an option because the processing chain leading to the speaker has constraints on the peak level. For example, the amplifier driving the speaker has peak voltage limits. A high input voltage can also damage the piezoelectric speakers.

In the telecommunication industry, it has long been a goal to have as much signal energy as possible for a given peak level (see, for example, [10]). This signal attribute is often expressed as the *peak-to-RMS ratio*, or *crest factor*,

$$\mathrm{CF}(t) = \frac{|x(t)|_{\mathrm{peak}}}{x_{\mathrm{RMS}}(t)},$$

where $|x|_{\mathrm{peak}}$ is the local maximum of the envelope modulating x, and x_{RMS} is the local root-mean-square value of x, approximating its short-time average power. Typical speech sentences have a peak-to-RMS ratio of 10 to 14, and unprocessed music recordings 8 to 10. When applied to audio signals, peak-to-RMS reduction algorithms seek to provide useful amounts of compression with minimum degradation of the sound quality. For example, Quatieri and McAulay applied the peak-to-RMS reduction approach based on a sinusoidal model to enhance speech signals transmitted by AM broadcasting [83].

Dynamic range compression (DRC) is a related method that employs fast-acting automatic gain control (AGC) to maintain the signal level near the peak at all times. It improves the peak-to-RMS ratio by equalizing the signal—amplifying the quiet portion and attenuating the high-energy portion. However, when it is directly applied to an audio signal, DRC tends to apply a rapidly varying gain, which causes undesirable signal modulation [4]. To cancel this artifact, the sub-band processing scheme can be used. It was demonstrated that effective DRC can be done in constant-Q sub-bands by using a band-limited gain control [16]. This approach works because the DRC gain can vary more rapidly in the high-frequency bands than in the low-frequency bands. A similar development on multi-channel DRC for music signals is presented by Schmidt and Rutledge [95].

2.4 Algorithm Implementation

The sound enhancement system shown in Figure 1 can be realized in computer simulation and in real-time hardware. The implementation of the components is discussed below.

The filter bank is composed of twenty-seven 1/3-octave 2nd-order Butterworth band-pass filters, whose center frequencies range from 50 Hz to 19 kHz. They approximate the critical bands in some models of the auditory system.

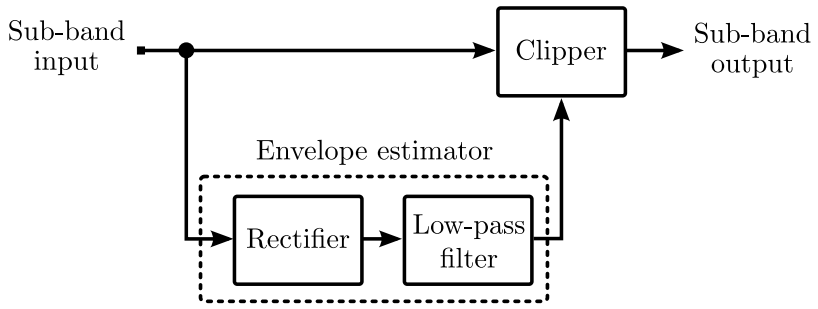


Figure 4: The clipper nonlinearity to create higher harmonics in the bass extension component. The envelope of the sub-band signal is extracted using a full-wave rectifier followed by a low-pass filter. The sub-band signal is then clipped at half of the envelope level.

The bass extension component is depicted in Figure 4. It uses a clipper nonlinearity to generate higher harmonics from sub-band signals in the low frequencies. The clipper has a transfer function

$$x_{\text{out}}(t) = \begin{cases} l_c & \text{if } x_{\text{in}}(t) > l_c, \\ x_{\text{in}} & \text{if } |x_{\text{in}}(t)| \leq l_c, \\ -l_c & \text{if } x_{\text{in}}(t) < l_c, \end{cases} \tag{1}$$

where l_c is a positive constant that determines the clipping level. The clipper generates odd harmonics for pure sinusoids [58]. To accommodate for a wide range of signal level, we set this clipping level to 50% of the *local* signal level, which can be estimated by the signal envelope. In this implementation, the envelope of the sub-band signal is extracted by low-pass filtering the rectified signal. The sub-band signal is then clipped at half of the envelope level.

The dynamic range compression component is depicted in Figure 5. It first extracts the envelope of the sub-band signal. From the sub-band envelope, it finds the local peak and local RMS value. In this implementation, each "local" frame contains 2500 samples, about 60 ms for CD-quality audio. The crest factor is then computed and passed through an empirically determined gain-limiting function. Shown in Figure 6, the gain-limiting function prevents the signal from being excessively boosted. It is almost linear for a gain (in linear scale) less than 3 and slowly saturates at 4 to prevent perceivable signal distortion. Finally, the computed gain is multiplied with the sub-band signal.

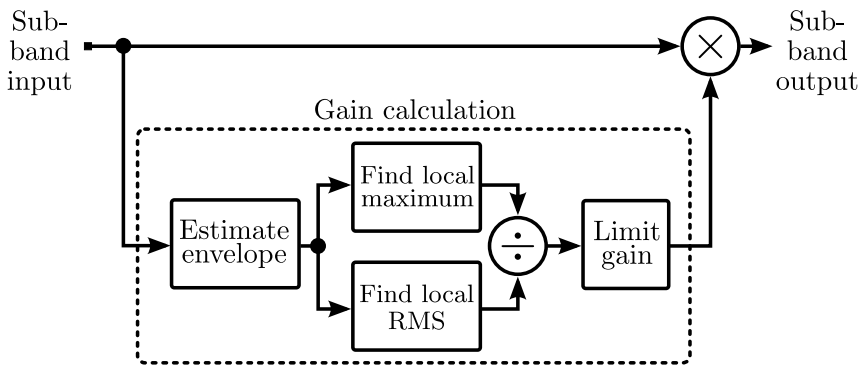


Figure 5: Sub-band dynamic range compression to reduce peak-to-RMS ratio. First, the local peak and RMS value of the sub-band signal is computed from its envelope. The peak-to-RMS ratio is then processed by a gain-limiting function (shown in Figure 6). The gain is applied to the sub-band signal.

2.5 Experimental Results

The entire system was first simulated in MATLAB. It was then implemented on a floating-point DSP board for evaluation and subjective testing. The experimental setup is illustrated in Figure 7.

A subjective test was conducted to evaluate the performance of the proposed system. The subjects were presented with five stimuli, which contained four songs and one artificially constructed chirp signal (1 to 1000 Hz). The subjects controlled the DIP switches on the DSP board to activate three processing schemes, which were unknown to the test subjects:

- unprocessed,
- bass extension only (BWE),
- sub-band DRC only (DRC), and
- bass extension and sub-band DRC (BWE+DRC).

The subjects were then asked to rank the processed signals in terms of "richness" (giving the sensation of being generated by a larger speaker) and overall sound quality. They did not rank the individual stimuli; instead, they were instructed to provide an average rank for all five stimuli.

We had eleven test subjects, all graduate students with normal hearing, in the experiment. The experimental results are presented in Table 1. The results show the number

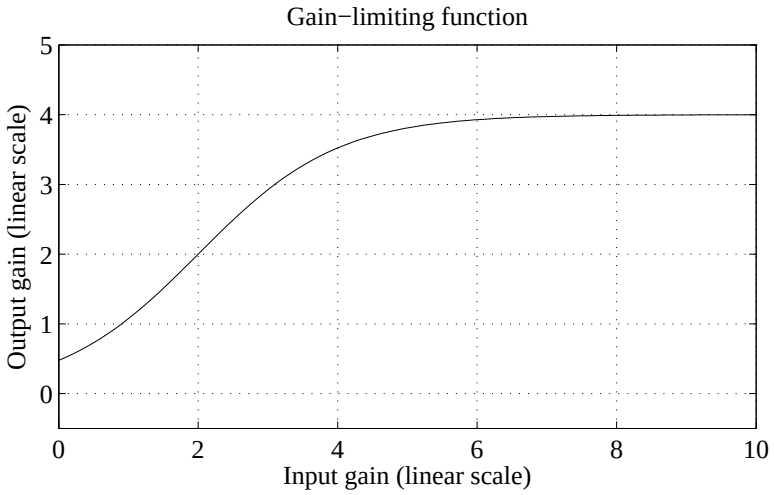


Figure 6: The empirically determined gain-limiting function in the sub-band dynamic range compression block. It is modified from a sigmoid function. The function is almost linear for a gain (in linear scale) less than 3. It then slowly saturates at 4 because a higher gain would distort the audio signal.

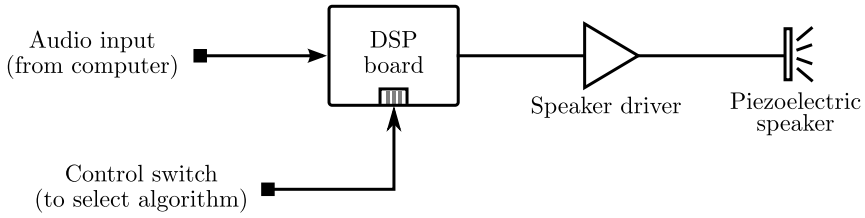


Figure 7: Evaluation and experimental setup. The algorithm is programmed in a floating-point DSP board (Texas Instrument TMS320C6713). The amplifier is provided by National Semiconductor, and the piezoelectric speaker is manufactured by muRata.

of times the algorithm is top-ranked. If two algorithms are tied for number one, both are counted as top-ranked. It is observed that most of the subjects consider BWE+DRC gives the "richest" output. However, the test subjects consider both DRC and BWE+DRC give the best overall quality.

2.6 Discussion

The proposed system combines two sound enhancement techniques to improve the performance of piezoelectric loudspeakers. The subjective test results suggest that the proposed system works well for both the "richness" and overall quality criteria. In particular, it gives the best result in terms of giving the sensation of a larger speaker. However, when the

Table 1: Experimental results from the subjective test. The numbers correspond to the number of times the algorithm is ranked best in the "richness" and overall quality criteria. If two algorithms are tied in top, they are both counted. If no algorithm is ranked number one, the result is omitted.

Algorithm	"Richness"	Overall quality
Unprocessed	0	0
BWE	1	1
DRC	2	6
BWE+DRC	9	5

overall sound quality is considered, the proposed system works as well as sub-band dynamic range compression alone.

Some subjects comment that even though the inclusion of the psychoacoustic bandwidth extension algorithm boosts the low-frequency sound level, it also introduces perceivable distortion for some music. The distortion produced by bass extension is most obvious in sounds that inherently contain little distortion, such as classical music. Conversely, bass extension improves the overall result when it is applied to sounds with complex structure, such as techno and rock music.

CHAPTER III

AUDIO POWER REDUCTION FOR LOUDSPEAKERS

Reducing the power consumption in loudspeakers improves the overall power-efficiency of portable electronic devices, such as media players, mobile phones, and hand-held game consoles. Power management also prevents the loudspeakers from overheating, hence providing thermal protection. Previous work in speaker power management has taken an adaptive gain control (AGC) approach. They typically infer or measure the current applied to the speaker and reduce the gain if the current exceeds a certain threshold [72]. Applying a negative gain leads to a perceived difference in the volume. Furthermore, because we perceive loudness differently at different frequencies, blindly reducing the gain can lead to perceptual degradation of the sound.

There are two aspects of human audio perception that we can exploit to solve these sound-quality issues. First, the human auditory system does not perceive sound levels equally across all frequencies [1]. The "equal-loudness" contour for humans is often quantified by the Fletcher-Munson curves. Second, when multiple sound stimuli are presented to a human observer, only some of the stimuli is perceived. This phenomenon is used in lossy audio coding, in which quantization errors (as a byproduct of encoding) are restricted such that the introduced noise is masked by the signal [14]. Applying these psychoacoustic principles, we can modify and even remove parts of a sound signal without changing its perceived loudness or sound quality. In this chapter, we present an algorithm to reduce loudspeaker power consumption without significantly degrading the sound quality. The overall architecture of the proposed system is depicted in Figure 8. This scheme works well for all loudspeakers, but it is especially effective for small speakers used in portable consumer electronic devices because small speakers typically distort the audio signal. As a result, it is easy to "hide" the artifacts of a pre-processed signal.

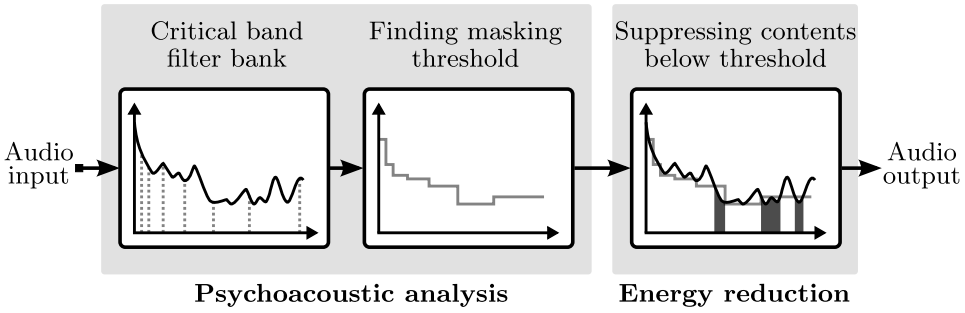


Figure 8: Overall architecture of the proposed perceptual audio compression system for reducing the power consumption in loudspeakers.

3.1 Power Management in Small Speakers

As portable electronic devices become smaller and ubiquitous, battery life is of great concern. Reducing the power spent in driving small speakers in portable electronics would prolong the battery life for a variety of devices.

Thermal protection is a concern for electrodynamic speakers because overheating can cause irreparable damage to the speaker elements. Methods to protect against overheating use temperature and current measurements to identify overheating conditions and adaptively reduce the gain to the amplifier to prevent the thermal damage [51]. Attenuation is applied uniformly across all frequencies. This makes the sound quieter, thus saving power.

3.2 Perceptual Audio Energy Reduction

Theories of audio coding are based on psychoacoustic principles, which characterize how humans perceive sounds. Modern audio codecs, such as the popular MP3, AAC, and RealAudio formats, utilize some form of a perceptual coding algorithm. These algorithms take advantage of the fact that not all auditory information is perceived equally by an "average human observer," and one can remove the irrelevant or redundant information without noticeably degrading the quality.

The compression algorithm presented in this chapter employs the same psychoacoustic principles used in popular audio formats. However, the motivation is fundamentally different. While most encoding compression standards aim to transparently reduce the storage space required by audio signals, our proposed system aims to reduce the output power of

audio signals while maintaining the perceived loudness level and sound quality. The two paradigms are henceforth referred to as *audio coding* and *power reduction*, respectively.

In both audio coding and power reduction, one estimates a hearing threshold, often expressed as a function of frequency; it is commonly assumed that signals below the threshold are inaudible. Although the popular audio coding schemes remove some portion of the signal below this level (especially in the high-frequency region), the hearing threshold is mainly used for bounding the quantization noise [14, 52]. In other words, the coder masks the undesirable artifacts of quantization by constraining the quantization noise to be below the masking threshold. In the proposed power reduction scheme, we suppress any signal content below the hearing threshold, hence reducing the total energy of the audio signal.

Perceptual audio power reduction can be used in any loudspeakers, but there are additional benefits in applying the scheme to small speakers because they typically have narrower bandwidth and higher distortion [59]. The narrow bandwidth means that a portion of the signal applied to the amplifier and loudspeaker is not transduced into sound energy, and the high distortion makes suppressing signals below threshold straightforward. Therefore, we reduce the energy of the audio signal in such a way that the processing is unnoticeable in small loudspeakers.

3.3 Implementation

In the audio power reduction algorithm, the key is finding the masking threshold. The processing follows the following steps:

1. Analyze the signal in the critical bands;
2. find the masking thresholds;
3. suppress the signal below the threshold curve; and
4. combine the processed sub-band signals.

For each 10-ms time-frame, the signal is divided into 25 critical bands in the frequency domain. This operation models the band-pass filtering effect of the cochlea. The power

spectrum of each band is computed using

$$P(\omega) = \Re\{f(\omega)\}^2 + \Im\{f(\omega)\}^2. \tag{2}$$

Then we sum the energy in each critical band to obtain the discrete Bark spectrum

$$B(n) = \sum_{\omega=bl_n}^{bh_n} P(\omega), \tag{3}$$

where bl_n and bh_n are, respectively, the lower and higher boundaries of the n-th critical band; the Bark spectrum is a subjective measure of loudness. To compensate for simultaneous masking (i.e., a sound makes another sound inaudible), a spreading function is convolved with the discrete Bark spectrum to generate

$$C(n) = B(n) * SF(n), \tag{4}$$

where the spreading function is empirically measured and reported in the literature (Figure 9(a)). To account for the different masking effects rendered by noise-like and tone-like signals, a *coefficient of tonality* is derived

$$\alpha = \min\left\{\frac{\mathrm{SFM_{dB}}}{-60}, 1\right\}, \tag{5}$$

where SFM is the spectral flatness measure and is given by

$$\mathrm{SFM} = \frac{\mu_g}{\mu_a}. \tag{6}$$

Here, the numerator and denominator are the geometric and arithmetic means of each Bark spectrum band, respectively. The offset (in dB) from the Bark spectrum is computed as

$$O(n) = \alpha(14.5 + n) + 5.5(1 - \alpha), \tag{7}$$

and the threshold from the Bark spectrum is given by

$$T(n) = 10^{\log_{10}\{(C(n)\} - \frac{O(n)}{10}}. \tag{8}$$

The *absolute threshold of hearing*, which is depicted in Figure 9(b), must also be taken into account. Finally, the masking threshold is given by

$$T_m(n) = \max\{T_q(n), T(n)\}. \tag{9}$$

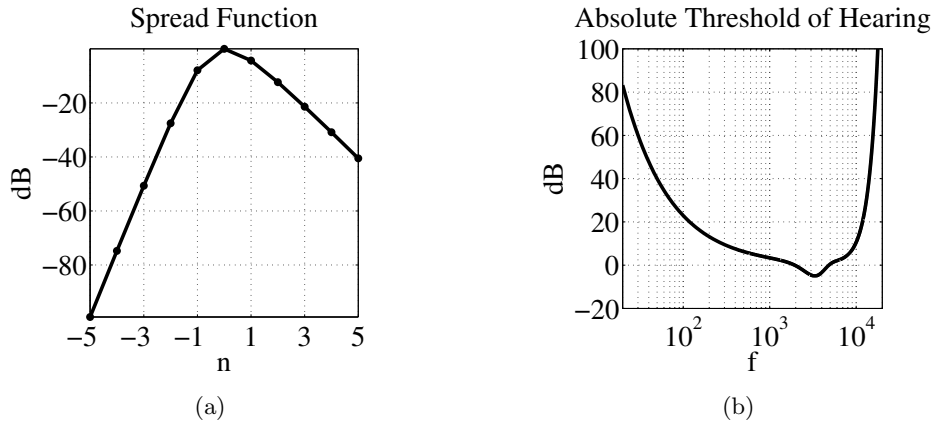


Figure 9: Functions used in the masking threshold calculation [77]. (a) Spread function to account for simultaneous masking. (b) Absolute threshold of hearing to represent the lowest audible sound.

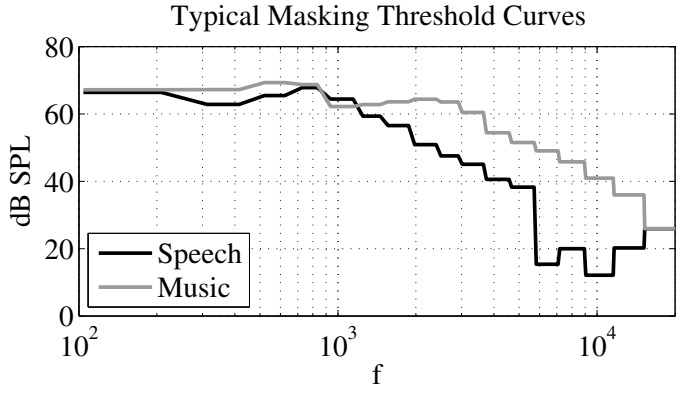


Figure 10: Typical masking threshold curves for music and speech signals.

The psychoacoustic principles behind the aforementioned threshold-finding process is explained in [77]. Typical threshold curves for music and speech signals are illustrated in Figure 10.

After finding the masking threshold, we compare the spectrum of the original signal with the threshold. If a spectral component is below the masking threshold, it would be inaudible and can be suppressed or removed. Therefore, we can remove the inaudible components. In practice, suppressing the below-threshold components by 6 dB is sufficient to reduce the signal energy. Attenuating the signal is less aggressive than setting it to zero, and it causes less perceptible artifacts that might result from abrupt onsets and offsets of the spectral components.

3.4 Experimental Results

3.4.1 Objective Testing

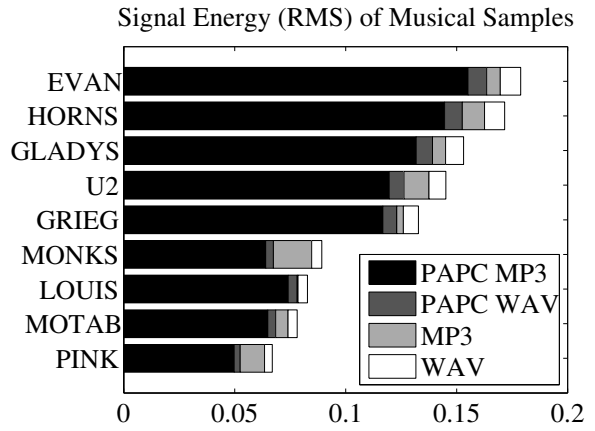
To determine the power savings from our method, we measured the root mean square (RMS) energy of the signals to be sent to the speaker. Since lossy encoding alters the energy and spectral density of the signal, we calculated RMS energy for the original signal (in lossless WAV format), the MP3-encoded (information-compressed) signal, the power-reduced WAV signal, and the power-reduced MP3 signal. Calculations were performed for both music (including a variety of genres with instrumental, vocal, and mixed examples) and speech.

Encoding to MP3 reduces the RMS energy by about 5% for all signals. Our proposed algorithm reduces the energy anywhere from 13% to 36% for music and 30% to 46% for speech, regardless of whether the source input is the original or encoded signal. To demonstrate that this energy is not limited to low frequencies (which cannot be reproduced by small speakers), we used a second-order Butterworth high-pass filter with a 500-Hz cut-off frequency to reduce the low-frequency components, and we measured the RMS energy of the high-passed signals again. In the music samples, we still demonstrated a 7% reduction (Figure 11(a)); in speech, 15% (Figure 11(b)).

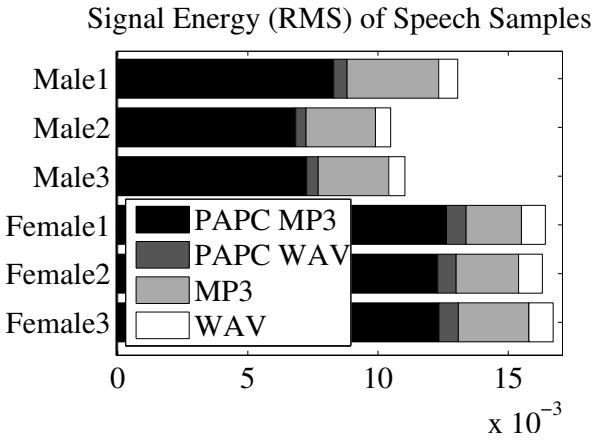
Greater energy reduction can be achieved for complex musical sounds while simple piano and soloist vocal sources allowed for less reduction. This result makes sense because most of the energy of latter signals is restricted to the most sensitive region of human hearing. Speech signals change slowly, allowing more aggressive energy removal without introducing "musical noise."[1] In all cases, our algorithm reduces the power more than MP3 encoding does, and MP3-encoded signals could be power-reduced by approximately the same ratio as the original signals.

Loudspeaker efficiency can vary widely by device—this is especially true of capacitive piezoelectric speakers. Since loudspeaker efficiency varies by frequency, we analyzed the frequency spectral density (calculated by Welch's method) to determine where the energy was removed. Most of the energy reduction occurs at the lowest and highest frequencies,

[1]The term *musical noise* refers to a processing artifact that contains spurious spectral peaks and sounds like fluctuating tones.



(a)



(b)

Figure 11: Root mean square (RMS) energy of the high-pass filtered signals for (a) musical samples and (b) speech samples. Perceptive audio power compression (PAPC) represents the processed signal. For music and speech, there is a similar power reduction for WAV (lossless) and MP3 (lossy) signals.

since human hearing is less acute at these frequencies, and some quantization noise was noticeable at high frequencies. The method gives some energy reduction at all frequencies. While the loudspeaker response will vary by frequency, spectral density reduction at all frequencies implies a power reduction independent of speaker characteristics, though the actual power reduction will vary by amplifier and loudspeaker [15].

3.4.2 Subjective Testing

Our power reduction algorithm generates high-quality audio signals. The processed signals are quite acceptable even when they are played back on high-quality audio equipments,

so this method works extremely well in general. Our main goal, however, is to generate signals that are indistinguishable from the original when played back-to-back. Therefore, we conducted a subjective test to evaluate the imperceptibility of the power reduction algorithm on small loudspeakers.

The subjects were presented with pairs of stimuli and were asked to decide whether the stimuli "sound the same" or "sound different." A total of 16 pairs of stimuli, with equal number of music and speech clips, were presented to each subject. Half the stimuli-pairs consist of pre- and post-processed signals; the other half consist of the same stimulus (either pre- or post-processed signal) as a control set. The permutation of stimuli-pair presentation was randomized, and the order of each pair of stimuli was also randomized.

The sound clips were encoded in lossless WAV format. The stimuli were played back on a small loudspeaker through an off-the-shelf audio driver board. The piezoelectric speaker has a low-frequency cut-off at around 800 Hz. Since the hearing threshold is a function of the distance between the listener and the speaker, the subjects were instructed to maintain a distance of 0.5 to 1 meter from the speaker.

We had 12 test subjects, all graduate students with normal hearing, in the experiment. The experimental results for music and speech are presented in Table 2 and Table 3, respectively. The results show a matrix of "stimuli presented" versus "stimuli perceived." For the music samples, when the subjects were presented with different stimuli, about half could not distinguish between the original signal and power-reduced signal. The result from the control group shows that when the same stimulus was presented, a number of subjects perceived a difference. For speech samples, most subjects could not distinguish the original signal and power-reduced signals, and the control group shows consistent results.

3.5 Discussion

An algorithm to reduce audio power is presented in this chapter. The algorithm estimates the masking threshold of an audio signal and suppresses any spectral content below the hearing threshold. The psychoacoustic processing introduces minimum noticeable artifacts in small loudspeakers.

Table 2: Subjective testing results for music. When the subjects were presented with different stimuli (a selection of musical samples from different genres), about half could not distinguish between the original signal and power-reduced signal. However, when the same stimulus was presented, a number of subjects perceived a difference.

	Perceived different stimuli	Perceived the same stimulus
Presented different stimuli	60%	40%
Presented the same stimulus	33%	67%

Table 3: Subjective testing results for speech. Both male and female speech samples were used in the experiment. Most subjects could not distinguish the original signal and power-reduced signal.

	Perceived different stimuli	Perceived the same stimulus
Presented different stimuli	38%	62%
Presented the same stimulus	15%	85%

Objective tests show a reduction of signal RMS energy; observing spectral density, we observe a reduction across all frequencies. This suggests power reduction for all loudspeakers regardless of their characteristics. In the subjective tests, taking the control group results into account, we observe that most subjects cannot distinguish between the pre- and post-processed signal, and the processing is especially "transparent" in speech signals. The loudness also appears to remain constant. Therefore, the processed signals are close to indistinguishable from the originals with differences that are only marginally significant.

This method suggests areas for future research. When energy was removed aggressively, there was musical noise apparent in the signal. This may be ameliorated by preventing rapid changes to the spectral attenuation in time, thus allowing more aggressive compression. We are also exploring specializing the algorithm for speech and music signals.

CHAPTER IV

SOUND DETECTOR AND CLASSIFIER

A pattern recognition system takes in raw data and takes an action based on the "category" of the pattern [29]. A sound classifier is a specialized pattern recognition system that works on signals in the audible frequency range. It takes an audio signal at the input and assigns the signal into different classes based on characteristics of the input signal. As depicted in Figure 12, sound classifiers typically contain two components: *feature extraction* followed by *classification*. In the feature extraction stage, the input audio signal is represented in distinguishable forms. The set of extracted features is known as the feature vector. In the classification stage, the feature vector is analyzed, and the input signal is classified into different categories.

4.1 Audio Features

An effective feature extractor yields distinct representations for different classes of input signals and therefore simplifies classification. Good audio features measure different values for sounds in different categories and maintain similar values for those in the same category [29]. In this section, we survey the audio features commonly used in sound classification.

Audio features are generally extracted in two levels: short-term "frame" level and long-term "clip" level [24]. Short-term features usually have a length of tens of milliseconds,

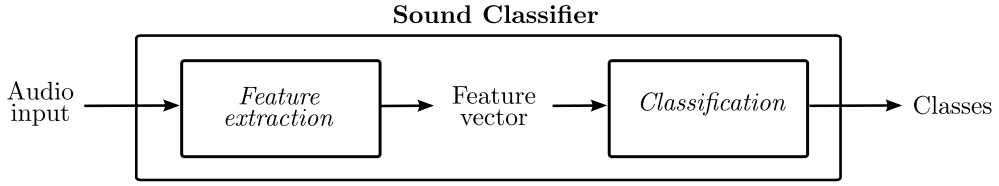


Figure 12: General architecture of a sound classifier. The system takes an audio signal as its input; it extracts distinguishable features from the input signal and constructs a *feature vector*; the feature vector is fed into a classification stage, where the signal is assigned into different categories or classes.

and it is assumed that the audio signal is stationary throughout this time. Long-term features cover a longer interval, and they are believed to reveal semantic meanings of an audio signal. In the proposed research, we investigate the classification of simple sounds and extremely short phrases whose semantic meanings are insignificant. As a result, we focus on the short-term audio features.

4.1.1 Energy-Based Features

In the discrete-time domain, the energy of a signal over a short time can be computed by

$$E_{\text{disc}} = \frac{1}{N} \sum_{n=0}^{N-1} x^2(n), \tag{10}$$

where E_{disc} is the signal energy for a discrete-time signal and N is the number of samples in the time frame. Similarly, in the continuous-time domain,

$$E_{\text{cont}} = \frac{1}{T} \int_0^T x^2(t)\ \mathrm{d}t, \tag{11}$$

where E_{cont} is the signal energy for a continuous-time signal and T is the length of the time frame. The signal is often decomposed into a plurality of frequency bands by a filter bank, and the energy of each band is computed. This is referred to as the *sub-band energy*.

Energy-based features include:

- *Modulation energy at 4 Hz*—Speech signals have a characteristic energy modulation peak around the 4-Hz syllabic rate [92].
- *Percentage of "low-energy" frames*—The proportion of frames with power less than half of the mean root-mean-square (RMS) power within a time window [92].
- *Low short-time energy ratio*—It is defined as the ratio of the number of frames in which the short-time energy is less than half of the average short-time energy in a time window. The ratio measures the variation of the short-time energy [65].
- *Voice-to-white energy ratio*—Energy of the human voice typically resides within the so-called "speech band" (30 Hz to 4 kHz). This feature measures the ratio of the energy of voice to that of the entire audible band (20 Hz to 20 kHz) [3].

4.1.2 Features Based on Zero-Crossings Rate

Zero-crossing rate is a useful quantity in the classical theory of random signal analysis [54]. This feature measures the number of zero-crossings per unit time. In the discrete-time domain, it is represented as

$$\mathrm{ZCR} = \sum_{n=1}^{N-1} |\mathrm{sgn}\left(x(n)\right) - \mathrm{sgn}\left(x(n-1)\right)|, \tag{12}$$

where ZCR is the zero-crossing rate in N samples within a time frame and $\mathrm{sgn}(\cdot)$ denotes the signum function. It is similarly defined in the continuous-time domain. The zero-crossing rate of a filtered signal is studied in the context of "dominant frequency" [54]. *Sub-band zero-crossing rate* is also used as a feature in audio classification [24]. A derivative feature is the *high zero-crossing rate ratio*; it is defined as the ratio of the number of frames in which the zero-crossing rate is higher than the average rate [65].

4.1.3 Spectrum-Based Features

The spectral density characterizes the frequency content of a signal. The power spectral density of a discrete-time signal (assuming it can be modeled by a wide-sense-stationary process) can be computed by using

$$S_x(f) = \mathrm{DTFT}\left\{r_x(k)\right\} = \sum_{k=-\infty}^{\infty} r_x(k) \cdot \mathrm{e}^{-j2\pi kf}, \tag{13}$$

where

$$r_x(k) = E\left\{x(n)x^*(n-k)\right\} \tag{14}$$

is the autocorrelation of x. In practice, the discrete-time Fourier transform (DTFT) is approximated by the discrete Fourier transform (DFT). Moreover, if we model the signal as a variance-ergodic process, then the autocorrelation can be estimated by the ensemble average of the process $y_k(n) = x(n)x^*(n-k)$ [48]

$$\hat{r}_x(k) = \sum_{n=0}^{N-1} y_k(n) = \sum_{n=0}^{N-1} x(n)x^*(n-k). \tag{15}$$

The continuous-time version is similarly derived [78]. Audio features based on spectral power include:

- *Spectral centroid*—The centroid is the "balance point" or "center of mass" of the power spectral density and, percetually, it is used to characterize the "brightness" of the sound [24, 92]. It is defined as

$$\mathrm{SC} = \frac{\sum_{n=0}^{N-1} n S_x(n)}{\sum_{n=0}^{N-1} S_x(n)}, \tag{16}$$

  where S_x is the discrete power spectral density in N bins (cf. definitions in [60, 63]).

- *Bandwidth*—This feature conveys the spread of energy with respect to the spectral centroid [24]. The bandwidth is computed as the magnitude-weighted average of the differences between the spectral components and the spectral centroid [109]

$$\mathrm{BW} = \sqrt{\frac{\sum_{n=0}^{N-1} (n - \mathrm{SC})^2 S_x(n)}{\sum_{n=0}^{N-1} S_x(n)}}, \tag{17}$$

  where SC is the spectral centroid defined in Equation 16 and S_x is the discrete power spectral density in N bins.

- *Spectral flux*—It evaluates the change in spectrum magnitude, and is defined as the ℓ_2-norm of the frame-to-frame spectral amplitude difference [24, 65, 92].

4.1.4 Mel-Frequency Cepstral Coefficients (MFCCs)

Mel-frequency cepstral coefficients (MFCCs) are a feature set that exploits the front-end filter bank model of the auditory system; they are widely used in speech and speaker recognition. The *cepstrum* can be loosely defined as the spectrum of the log-spectrum of a signal [22, 75]. Numerically, it can be computed by the following three steps:

$$\begin{aligned} X(k) &= \textstyle\sum_{n=0}^{N-1} x(n) \mathrm{e}^{-j\frac{2\pi}{N}kn}, \\ \hat{X}(k) &= \log\left(X(k)\right), \\ \hat{x}(n) &= \tfrac{1}{N} \textstyle\sum_{n=0}^{N-1} \hat{X}(n) \mathrm{e}^{j\frac{2\pi}{N}kn}, \end{aligned} \tag{18}$$

where $\hat{x}$ is referred to as the (complex) discrete cepstrum [82]. The *mel*[1] scale approximately translates the frequency axis into a psychoacoustic pitch scale. The mapping is linear for frequencies below 1000 Hz and logarithmically increases with higher frequency. This can be accomplished by weighting the spectrum by an overlapping, triangular mel-scale filter bank [27]. There are many versions of MFCCs; the following is a common derivation:

[1] "Mel" refers to "melody"; it denotes the sense of pitch as perceived by an average human listener.

1. Take the Fourier transform of a windowed signal.
2. Convert the energy spectrum to the mel scale by applying the mel-scale filter bank.
3. Take the logarithm of the mel-frequency energy coefficients.
4. Take the discrete cosine transform (DCT) of the log-domain energy coefficients.

The resulting coefficients are used in sound classification [3, 24]. In addition to MFCCs, other cepstrum-based features, such as the *cepstrum resynthesis residual magnitude*, have been used [92].

4.1.5 Features Based on Linear Predictive Coding (LPC)

Linear predictive coding (LPC) is widely employed in discrete-time speech processing for representing speech signals in a compact form. Speech samples are encoded as a linear combination of the previous samples. In a basic speech production model, over a single glottal cycle the vocal tract is embodied by an all-pole transfer function (in z-domain)

$$\begin{aligned} H(z) &= AG(z)V(z)R(z) \\ &= \frac{A}{1 - P(z)} \qquad (19) \\ &= \frac{A}{1 - \sum_{k=1}^{p} a_k z^{-k}}, \qquad (20) \end{aligned}$$

where A is the gain, G is the glottal flow, V is the vocal tract transfer function from the volume velocity at the glottis to the volume velocity at the lips, R represents the radiation load at the lips, and P is a p^{th}-order predictor polynomial. The input to this system is either an impulse train or noise excitation. The prediction coefficients (a_ks in Equation 20) can be estimated by minimizing the error between actual and predicted speech signals. LPC and its related representations (e.g., partial correlation coefficients) are often used in low-bit-rate speech coding, but they can also be used to estimate basic speech parameters such as pitch and formants [24]. Therefore, important audio features can be extracted from LPC analysis. Some of the LPC-based features include:

- *Line spectral frequencies (LSFs)*—In Equation 19, the term $(1 - P(z))$ can be decomposed into

$$\begin{cases} \hat{P}(z) &= [1 - P(z)] + z^{-(p+1)} \left[1 - P\left(z^{-1}\right)\right], \\ \hat{Q}(z) &= [1 - P(z)] - z^{-(p+1)} \left[1 - P\left(z^{-1}\right)\right]. \end{cases} \tag{21}$$

  The LSFs are the roots of $\hat{P}$ and $\hat{Q}$. They are shown to demonstrate explicit differences across audio classes [30, 65].

- *Linear prediction zero-crossing ratio*—It is defined as the ratio of the zero-crossing count of the input and that of the LPC analysis output. This feature is said to quantify the "correlation structure" of a sound (e.g., voiced vs. unvoiced speech) [30].

4.1.6 Features Based on Models of the Auditory System

Finding the audio features that are robust in noisy environments is a recent research topic. A bigger challenge is to find the audio features that are noise-robust and easy to compute; the latter is important in real-time applications. Toward this goal, researchers have drawn inspirations from the human auditory system. Humans are able to quickly detect and perceive sounds in vastly different listening environments, so it makes sense to adopt processing strategies inherent to the human ear.

Many models of the human auditory system exist—some are used in understanding the hearing organs; others are used in mimicking the signal processing path. A popular model that has inspired the development of noise-robust audio features is the *early auditory representation*[2] proposed by Yang et al. [111]. The authors described that they attempted to capture "the succession of linear, nonlinear, adaptive, and cross channel processing stages that are known to occur in the auditory system" with a minimal representation.

The early auditory model is shown in Figure 13. Conceptually, it is divided into three stages: the *analysis stage*, which consists of the cochlear filters; the *transduction stage*, which converts mechanical displacement into electrical signal in the inner hair cells; and the

[2]Computational auditory models are based on neurophysiological, biophysical, and psychoacoustical investigations at various stages of the auditory system. They generally contain two basic stages: An *early stage* models the transformation of the acoustic signal into an internal neural representation, and a *central stage* that analyzes the resulting representation [31]. The "early auditory representation" refers to modeling the early stage.

reduction stage, which limits the lateral interaction of the neurons. The following is a brief description of the operations in the three stages [24, 87].

1. *Analysis stage: cochlear filters*—An audio signal entering the ear can be viewed as a spatio-temporal pattern of vibration along the basilar membrane, and the cochlea has band-pass frequency responses for each location. The cochlear filters $h(t, s)$ typically have 20 dB/decade roll-off on the low-frequency side and a very sharp roll-off on the high-frequency side.

2. *Transduction stage: hair cells*—The ∂t-block converts the instantaneous membrane displacement into velocity by taking the temporal derivative. The nonlinearity of the ionic channel (through the hair cells) is modeled by a sigmoid-like compression function $g(\cdot)$. A low-pass filter $w(t)$ is then used to model the leakage of the cell membrane. The output of this stage represents the neural spikes in the auditory nerves.

3. *Reduction stage: lateral inhibitory network*—Lateral inhibition in the cochlear nucleus is modeled by a spatial derivative (the ∂s-block). This is followed by a spatial low-pass filter $v(s)$, which accounts for the finite spatial extend of the lateral interactions. The nonlinearity of the neurons is modeled by a half-wave rectifier (HWR). Finally, a temporal integral is used to simulate the central auditory neurons, which cannot react to rapid temporal modulations. The output of this stage is called the *auditory spectrum*.

From a sound classification perspective, the self-normalization property of the early auditory model is attractive. Using stochastic analysis, Wang and Shamma showed that the output of the model—the auditory spectrum—effectively computes a spectrum divided by a smoothed version of itself [106]. They also demonstrated that this property leads to robustness against scaling and noise corruption. The self-normalization property was later verified using closed-form expressions [25].

Ravindran et al. proposed the *noise-robust auditory features* (NRAF), which are based on a simplified model of the early auditory system, as shown in Figure 14 [87]. The cochlear

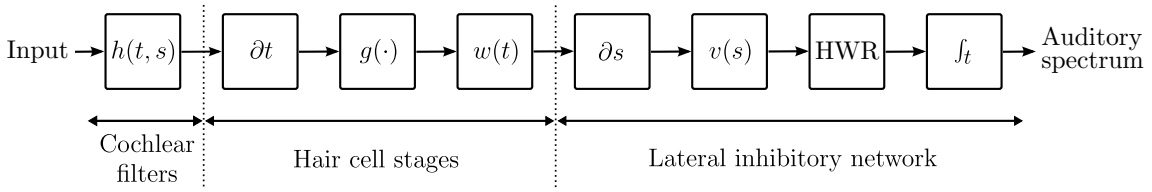


Figure 13: A mathematical representation of the early auditory system proposed by Yang, Wang, and Shamma [106, 111]. The model has three stages: the *analysis stage* that consists of the filters in the cochlea, the *transduction stage* that converts mechanical displacement into electrical activity in the inner hair cells, and the *reduction stage* that contains lateral inhibitory network in the cochlear nucleus.

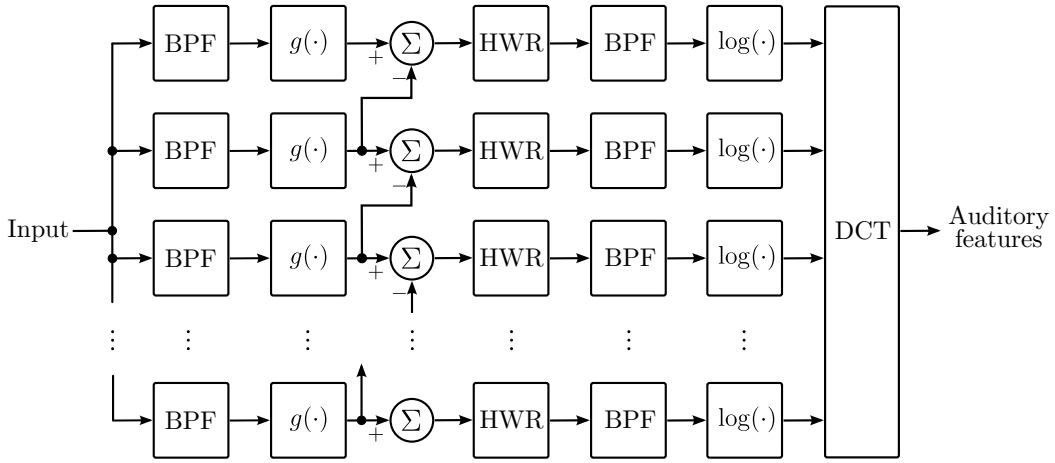


Figure 14: The noise-robust auditory features (NRAF) proposed by Ravindran, Schlemmer, and Anderson [87]. The system largely follows the processing blocks in Figure 13. The temporal low-pass filter $w(t)$ and spatial low-pass filter $v(s)$ are deliberately omitted because doing so constitutes a valid approximation at moderate sound intensities. The auditory spectrum is log-compressed and transformed using the discrete cosine transform (DCT), which effectively decorrelates the channels.

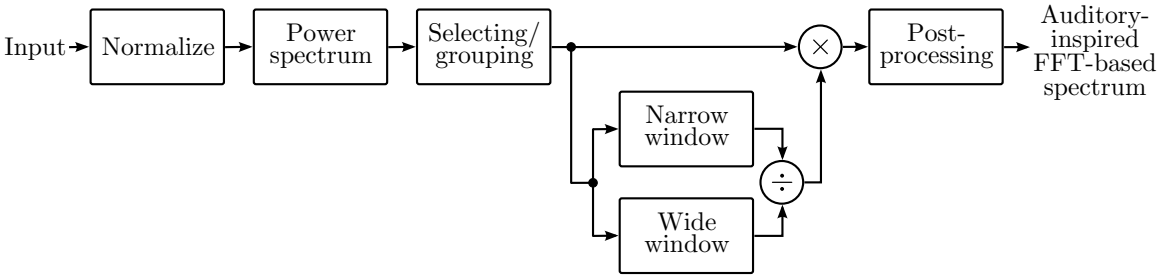


Figure 15: An FFT-based self-normalizing spectrum proposed by Chu and Champagne [25]. The signal is normalized by its RMS value. An FFT-based algorithm calculates the power spectrum of the signal. The resulting spectrum is sampled to form a power spectrum vector, which is then scaled by a self-normalization coefficient. The spectrum data is smoothed to form the FFT-based auditory spectrum.

filters are implemented through a band-pass filter bank. The nonlinearity of the ionic channel $g(\cdot)$ is implemented by a sigmoid-like function. The spatial derivative ∂s is approximated by a difference operation between adjacent frequency channels. The half-wave rectification stage is retained, and the temporal averaging is implemented by a low-pass filter. A logarithmic compression function is included to model outer hair cells. Finally, the log-compressed auditory spectrum is decorrelated by the discrete cosine transform (DCT).

Chu and Champagne derived an *FFT-based auditory spectrum* that allows audio feature extraction [25]. A schematic of the implementation is illustrated in Figure 15. The input signal is first normalized with respect to the square root value of its average energy. The short-time power spectrum is calculated by a windowed FFT algorithm. The power spectrum is then resampled to reduce the dimension of the spectrum vector. Local self-normalization is implemented through the use of a pair of narrow and wide filters; they correspond to a "fast" and a "slow" running average, respectively. The spectrum vector is scaled by a self-normalization coefficient, which is the ratio of the running averages. Finally, in the post-processing stage, the spectrum data is smoothed to generate the FFT-based self-normalizing auditory spectrum.

4.1.7 Other Audio Features

Other audio features reported in the literature include the following:

- *Pitch*—a perceptual term to represent the fundamental frequency [24];

- *Pulse metric*—an estimation of "rhythmicness" [92];
- *Entropy*—a measure of the noise-like or tone-like behavior of the signal [32].

In recent years, spike-based signal processing has become a popular research topic. Spiking signal representation mimics the voltage spikes that are ubiquitous in biological nervous systems. Spiking neuronal models are different from the early auditory model discussed in the previous section in that the spiking models directly employ the biological abstraction of individual neurons. These abstractions include the simple integrate-and-fire model, the more elaborate Hodgkin-Huxley model, and models for a population of neurons [36]. Abdalla and Horiuchi used the spiking representation to detect peak-periodicity. The system detects voiced segments in speech signals [2].

4.2 Pattern Classification

In the classification stage, the system uses the feature vector provided by the feature extraction stage to assign the input signal to a category. Because of the abstraction provided by the feature vector representation of the input data, classification methods are typically domain-independent. In other words, while the feature extraction stage is specifically designed for audio signals, a generic classification stage can be used for a variety of signals.

Classification is concerned with predicting a discrete variable, and there are a number of related disciplines. For example, *density estimation* predicts the probability density function; *regression* predicts a continuous variable, *machine learning* emphasizes using computational statistical methods to solve the aforementioned problems; *data mining* usually deals with enormous datasets. In this section, we survey some of the popular classification schemes; attention is paid to those suitable for analog implementation.

4.2.1 Predicting a Discrete Variable

The goal of classification is to obtain a model from the training data—this is the process of *learning*. Based on the model, the classification system predicts something about the data that it has not encountered before. An important assumption is that the test data comes from the same probability distribution as the training data [39]. Specifically, we can

consider independent and identically distributed (i.i.d.) data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \in \mathcal{X} \subset \mathbb{R}^D$ is a D-dimensional vector and y_i is in some finite set $\mathcal{Y}$. A *classification rule* is a function $h : \mathcal{X} \to \mathcal{Y}$. When a new x is observed, y is predicted to be $h(x)$.

We want to find a classification rule h that accurately predicts y from x. Toward this goal, we need to define a loss function for classification. The *true error rate* of a classifier h is [107]

$$L(h) \equiv \mathbb{P}\left\{h(X) \neq Y\right\}, \tag{22}$$

and the *empirical error rate* can be estimated by

$$\hat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} I\left(h\left(x_i\right) \neq y_i\right), \tag{23}$$

where I is the indicator function. If the classification system is a two-class detector, i.e., $\mathcal{Y} = \{0, 1\}$, then we can define the *discriminant function*[3]

$$r(x) = \mathbb{E}(Y|X = x) = \mathbb{P}(Y = 1|X = x). \tag{24}$$

From Bayes' theorem, we can derive that

$$r(x) = \frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_0 f_0(x)}, \tag{25}$$

where $f_1(x) = f(x|Y = 1)$ and $f_0(x) = f(x|Y = 0)$ are the conditional densities (or likelihoods); $\pi_1 = \mathbb{P}(Y = 1)$ and $\pi_0 = \mathbb{P}(Y = 0) = 1 - \pi_1$ are the priors. The *Bayes classification rule* is defined as

$$h^*(x) = \begin{cases} 1, & \text{if } \pi_1 f_1(x) > \pi_0 f_0(x), \\ 0, & \text{otherwise;} \end{cases} = \begin{cases} 1, & \text{if } r(x) > 1/2, \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

The Bayes classifier can be shown to be optimal, i.e., $L(h^*) \leq L(h)$ for any other classification rule h [107]. The set

$$\mathcal{D}(x) = \{x : \mathbb{P}(Y = 1|X = x) = \mathbb{P}(Y = 0|X = x)\} \tag{27}$$

[3]The discriminant function (or regression function in the continuous case) $r(x) = \mathbb{E}(Y|X = x)$ gives the best guess for y given a value of x.

is called the *decision boundary*. In the two-class case, the classifier considers all data points on one side of the decision boundary as belonging to one class and those on the other side as belonging to the other class. These results can be extended to K classes, i.e., $Y \in \mathcal{Y} = \{1, 2, \ldots, K\}$. In this case, the k-th discriminant function becomes

$$r_k(x) = \frac{f_k(x)\pi_k}{\sum_{i=1}^{K} f_i(x)\pi_i}, \, k = 1, \ldots, K, \tag{28}$$

and the Bayes classification rule is

$$h^*(x) = \arg\max_k \left\{ r_k(x) \right\}, \, k = 1, \ldots, K. \tag{29}$$

4.2.2 Generative Approach: Gaussian and Linear Classification Rules

One way to design a classifier is to estimate the conditional densities f_k in the discriminant function in Equation 28 and apply the Bayes classification rule in Equation 29. This approach of modeling the distribution of the input data is known as a *generative model* because it enables the generation of synthetic data points in the input space [12].

A simple way to estimate the densities is to assume a multivariate Gaussian model. Let

$$f_k(x) = \frac{1}{|\Sigma_k|^{1/2}(2\pi)^{D/2}} \exp\left\{ -\frac{1}{2} \left(x - \mu_k\right)^{\mathrm{T}} \Sigma_k^{-1} \left(x - \mu_k\right) \right\}, \, k = 1, \ldots, K, \tag{30}$$

where $\mu_k \in \mathbb{R}^D$, Σ_k is a $D \times D$ symmetric, positive definite matrix, and $|\cdot|$ denotes the determinant of the matrix. In other words, $(X|Y = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$. In this case, the discriminant function in Equation 29 becomes

$$r_k(x) = -\frac{1}{2} \left(x - \mu_k\right)^{\mathrm{T}} \Sigma_k^{-1} \left(x - \mu_k\right) - \frac{1}{2} \log |\Sigma_k| + \log \pi_i. \tag{31}$$

If we assume that the covariance matrices $\Sigma_k = \Sigma, \; \forall k$ (but otherwise arbitrary), then Equation 31 can be simplified to

$$r_k(x) = x^{\mathrm{T}} \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^{\mathrm{T}} \Sigma^{-1} + \log \pi_k. \tag{32}$$

This is known as the *linear discriminant analysis* because r_k and the resulting decision boundary are linear in x.

In general, the mean μ_k and covariance matrix Σ_k in Equation 30 can be estimated by various parameter estimation methods, such as the method of moments, maximum likelihood, and Bayesian estimation. Furthermore, the simple Guassian model can be extended to use a mixture of Gaussian; the resulting classification rule is known as the *mixture Bayes classifier*. Nonparametric methods, such as as kernel density estimation, can also be used to estimate the densities [39, 107].

4.2.3 Discriminative Approach: Generalized Linear Models

Estimating the conditional densities can be a computationally demanding process, and it requires a large amount of training data. In addition, the conditional densities may contain information that has little effect on the final classification results [12]. An alternative approach is to directly estimate the discriminant function $r(x) = \mathbb{E}(Y|X = x)$; it is equivalent to modeling the decision boundary. This is known as the *discriminative model*.

In the two-class case, where $Y \in \mathcal{Y} = \{0, 1\}$, the classification rule is given by

$$h(x) = \begin{cases} 1, & \text{if } r(x) > 1/2, \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

Here, the equation $r(x) = 1/2$ describes the decision boundary. A simple model for r is the *linear regression* model

$$Y = r(x) + \epsilon = \beta_0 + \sum_{i=1}^{D} \beta_i X_i + \epsilon = \mathbf{X}\boldsymbol{\beta} + \epsilon, \tag{34}$$

where $\mathbb{E}(\epsilon|X_1, \ldots, X_D) = 0$, $\mathbf{X} = [1, X_1, \ldots, X_D]$, and $\boldsymbol{\beta} = [\beta_0, \ldots, \beta_D]^{\mathrm{T}}$. The coefficients $\boldsymbol{\beta}$ can be found by minimizing the residual sums of squares

$$\mathrm{RSS} = (Y - \mathbf{X}\boldsymbol{\beta})^{\mathrm{T}} (Y - \mathbf{X}\boldsymbol{\beta}) . \tag{35}$$

It can be shown that the pseudoinverse of $\mathbf{X}$ provides the least square solution $\hat{\boldsymbol{\beta}}$ that minimizes the RSS [107]:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1} \mathbf{X}^{\mathrm{T}} Y. \tag{36}$$

Therefore, the discriminant function in Equation 33 is given by

$$\hat{r}(x) = \hat{\beta}_0 + \sum_{i=1}^{D} \hat{\beta}_i X_i. \tag{37}$$

The linear model can be generalized to take into account the discrete nature of the target classes. The *logistic regression* method[4] adopts a different model for the discriminant function

$$r(x) = p\left(\boldsymbol{\beta}\right) = \sigma\left(\beta_0 + \sum_{i=1}^{D} \beta_i X_i\right), \tag{38}$$

where σ is the logistic sigmoid function

$$\sigma(z) = \frac{\mathrm{e}^{\,z}}{1 + \mathrm{e}^{\,z}}. \tag{39}$$

Because the target is binary, each observed target Y_i can be modeled by the Bernoulli distribution

$$(Y_i | X_i = x_i) \sim \mathcal{B}(p_i), \tag{40}$$

where p_i is the "success" rate as given in Equation 38. The conditional likelihood function for a data set of size N is

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{N} p_i\left(\boldsymbol{\beta}\right)^{Y_i} \left(1 - p_i\left(\boldsymbol{\beta}\right)\right)^{1-Y_i}. \tag{41}$$

We can then use maximum likelihood to numerically evaluate the parameter $\boldsymbol{\beta}$ [12, 107].

Another generalized linear model is the *perceptron*, which tries to constructs linear decision boundaries by explicitly separating the data into different classes. It is so called because it can be seen as the simplest kind of artificial neural network (to be discussed in Section 4.2.4). In this model, it is more convenient to represent the binary target classes as $Y \in \mathcal{Y} = \{-1, 1\}$. A separating plane in the feature space is given by the equation

$$g(x) = \beta_0 + \sum_{i=0}^{D} \beta_i x_i = 0. \tag{42}$$

The goal is to find the separating plane such that its distances to all misclassified points are minimized. In this case, the classification rule is

$$h(x) = \mathrm{sgn}\{g(x)\}. \tag{43}$$

[4]Logistic regression is really a method for classification; it is so named because of its roots in regression.

Using the vector notation, $\beta^{\mathrm{T}}x = \sum_i \beta_i x_i$. If a target $y_i = 1$ is misclassified, then $\beta_0 + \beta^{\mathrm{T}}x_i < 0$, and vice versa. This simplifies to minimizing

$$D(\beta_0, \beta) = -\sum_{j \in \mathcal{M}} y_j \left(\beta_0 + \beta^{\mathrm{T}}x_i\right), \tag{44}$$

where $\mathcal{M}$ is the set of indices of misclassified data points. A number of methods are available to solve this optimization problem [29, 39, 47]. In general, the *perceptron learning algorithm* finds a separating plane if the training data are linearly separable, but it does not converge for nonseparable problems. The *least mean squares* (LMS) method always finds a plane to minimize (in the least-squares sense) misclassification, but it does not necessarily converge to the separating plane if the training data are linearly separable. The *Ho-Kashyap algorithm* strikes a balance between the two.

The generalized linear methods are well understood and easy to implement. Esmaili et al. employed linear discriminant function to perform content-based audio classification [32]; a commercially available software (SPSS) was used to train the classifer. Gestner et al. also applied linear discriminant function to detect glass break sounds [37]. The authors trained the classifier with the Ho-Kashyap procedure.

4.2.4 Neural Networks

Artificial neural networks (ANNs) originated in attempting to find mathematical representations of information processing in biological neural networks. They have been used in statistical modeling for pattern classification as well as in simulating neuronal signal processing in biology. The networks are made up of interconnecting artificial neurons that mimic some of the properties of biological neurons.

The basic neural network model is built from layers of artificial neurons, which can be described by a model similar to logistic regression [12]. We first construct M linear combinations of the input variables $x_1, \ldots, x_D$

$$a_j = b_j^{(1)} + \sum_{i=1}^{D} w_{ji}^{(1)} x_i, \; j = 1, \ldots, M, \tag{45}$$

where the superscript (1) denotes the first layer of the network. The parameters w_{ji} are referred to as *weights*, and b_j *biases*. The output quantities a_j are known as *activations*,

and they are transformed using a differentiable, nonlinear activation function h to give

$$z_j = h(a_j), \tag{46}$$

where the quantities z_j are called *hidden units*, and h is usually a sigmoid function, such as the logistic function in Equation 39 or the hyperbolic tangent function (tanh). The hidden units are again linearly combined to give the output unit activation

$$a_k = b_k^{(2)} + \sum_{i=1}^{M} w_{ki}^{(2)} z_i,\ k = 1, \ldots, K, \tag{47}$$

where K is total number of output units. Finally, the quantities a_k are transformed to give a set of network outputs

$$y_k = \sigma(a_k), \tag{48}$$

where σ is a nonlinear output activation function. Combining the expressions from various stages, a two-layer neural network takes the form

$$y_k(\mathbf{x}, \mathbf{w}, \mathbf{b}) = \sigma \left[b_k^{(2)} + \sum_{j=1}^{M} w_{kj}^{(2)} h \left(b_j^{(1)} + \sum_{i=1}^{D} w_{ji}^{(1)} x_i \right) \right], \tag{49}$$

where the inputs, weights, and biases are grouped into $\mathbf{x}$, $\mathbf{w}$, and $\mathbf{b}$, respectively. A two-layer neural network is pictorially shown in Figure 16. The model can be generalized to include additional layers that assume the form in Equations 47 and 48.

Multi-layer neural networks are more powerful than the linear models because the nonlinear activation functions at the hidden units allow them to approximate an arbitrary decision boundary. They can be trained by numerous methods, such as the well-known back-propagation algorithm [12, 29]. However, there is no "standard" way to design a neural network. Choosing a network architecture largely depends on the data representation and the application. Neural networks have been applied to many classification problems, including audio-related tasks. For example, Ravindran and Anderson utilized neural networks with physiologically inspired auditory features for audio classification [87].

4.2.5 Support Vector Machine

In the perceptron classification rule, the goal is to find a separating hyperplane such that its distances to all misclassified points are minimized. An alternative criterion is to find the

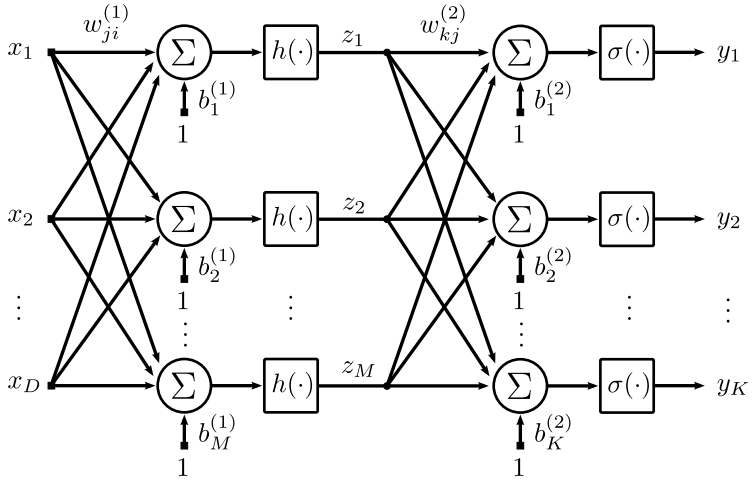


Figure 16: Illustration of a 2-layer artificial neural network. The network implements the equation $y_k = \sigma\left[b_k^{(2)} + \sum_{j=1}^{M} w_{kj}^{(2)} h\left(b_j^{(1)} + \sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)\right]$, for $k = 1, \ldots, k$, where h and σ are typically sigmoid functions.

hyperplane that maximizes the distance of the closest point from either class. This distance is called the *margin*, and points on the margin are know as *support vectors*. Support vector machines (SVMs) are a set of statistical learning methods that maximizes the margin.

For a linearly separable two-class data set ($Y \in \mathcal{Y} = \{-1, 1\}$), there are many hyperplanes that separate the two classes. The separating plane and classification rule are described by the Equations 42 and 43, respectively. We want to find a separating plane that creates the largest margin, which is given by the optimization problem [47]

$$\min_{\beta,\beta_0} ||\beta||, \text{ subject to } y_i\left(x_i^{\mathrm{T}}\beta + \beta_0\right) \geq 1, \; i = 1, \ldots. \tag{50}$$

This is a standard convex optimization problem; methods exist for finding the global optimum [39]. For practical problems, however, the data points are often not linearly separable. We introduce a *slack variable* ξ_i to the constraint, and solve the optimization problem

$$\min_{\beta,\beta_0} ||\beta||, \text{ subject to } \begin{cases} y_i\left(x_i^{\mathrm{T}}\beta + \beta_0\right) \geq 1 - \xi_i \;\; \forall i, \\ \xi_i \geq 0, \;\; \sum_i \xi_i \leq C, \end{cases} \tag{51}$$

where C is a tunable parameter that effectively bounds the total number of misclassifications.

SVMs have enjoyed popularity in recent years, and it has been applied extensively in audio classification. For example, Lin et al. classified and categorize audio data based on

wavelets and SVM [61]. Ravindran described an SVM-based cascade-of-classifiers structure in his dissertation [85].

4.3 Resampling and Model Combination

There are two criteria to evaluate a classification algorithm: bias and variance. The bias measures the accuracy, and the variance measures the precision of a classification method. The *bias-variance dilemma* states that procedures with increased flexibility to adapt to the training data (e.g., having more parameters) tend to have lower bias but higher variance [29]. Therefore, when we apply a classification algorithm to a new pattern recognition problem with unknown distribution, we need to estimate the bias and variance for the particular algorithm and data set. Two resampling methods, *jackknife* and *bootstrap*, can be used to estimate the statistics. In the jackknife method, one systematically recomputes the statistic estimate by leaving out one data point from the data set at a time. An estimate for the bias and variance is then computed from the "new" sets of observations. In the bootstrap method, one estimates the statistics by sampling with replacement from the original data set. The bias and variance estimates are then computed from the "bootstrap" data sets.

The idea of resampling can be extended to the design of classification algorithms. The method of *bagging* (contracted from "bootstrap aggregation") uses multiple versions of a training set with replacement. Each of these bootstrap data sets is used to train a different component classifiers; the final classification decision is based on the vote of each component classifier [29]. A related method that combines component classifier to form a strong aggregate classifier is known as *boosting*, in which successive component classifiers are trained with a subset of the training data that, given the current set of component classifiers, is prone to be misclassified. There are many variants of this scheme; the popular *AdaBoost* method is the focus of this thesis research. The AdaBoost algorithm is discussed in detail in Section 7.2. Other model combination methods include *stacking* and *Bayesian model averaging*.

4.3.1 Other Classification Methods

4.3.1.1 Nearest Neighbor

The *nearest neighbor* classification rule is an instance-based learning rule, which constructs hypotheses directly from the training instances. If we define the set $\mathcal{D}^n = (x_1, \ldots, x_n)$ to denote n labeled prototypes, then the nearest neighbor classification rule assigns a test point x to the prototype $x' \in \mathcal{D}^n$ that is closest to x [29]. The distance measure is usually the Euclidean distance. The nearest neighbor rule is suboptimal, but it is easy to implement. This classification method can be extended to the *k-nearest neighbor* classification rule, in which a decision is made by examining the labels on the k nearest neighbors and taking a majority vote.

4.3.1.2 Radial Basis Functions Networks

In the neural networks presented in Section 4.2.4, input variables are linearly combined and processed by an activation function. For example, the j-th hidden unit is computed by

$$z_j = h\left(b_j + \sum_{i=1}^{D} w_{ji} x_i\right), \tag{52}$$

where h is typically a sigmoidal activation function. Alternatively, a nonlinear radial basis function (RBF) can be used:

$$z_j = \phi\left(||\mathbf{x} - \mathbf{c}_i||\right), \tag{53}$$

where $\mathbf{x} \in \mathbb{R}^D$ is the input vector, $||\cdot||$ denotes the Euclidean norm, the function $\phi : \mathbb{R}^+ \to \mathbb{R}$ is the RBF, and $\mathbf{c}_i \in \mathbb{R}^D, i = 1, \ldots, D_r$ is the center vector of the RBF [21]. the Gaussian function

$$\phi(z) = \mathrm{e}^{-z^2/\beta},\ \beta > 0 \tag{54}$$

is commonly used. The hidden layers z_j are linearly combined to form the *radial basis function network*. The RBF can also be derived from a kernel machine framework [12]. In Section 5.2, an analog implementation of the RBF network is discussed.

4.3.1.3 Hidden Markov Model

A discrete random process takes on a state value at each time step. A Markov process is a memoryless discrete process in which the transition probabilities into the next state are dependent only on the current state. The *hidden Markov model* (HMM) is a Markov process in which every state generates an observation at each time step [39]. The HMM is widely used in speech recognition, but it has also been used in sound classification [114]. An HMM is defined by three parameters: the initial state probabilities π, the state transition probabilities matrix A, and the emission probability matrix B (denoting the probability of the states emitting observable symbols). Given an HMM $\theta(A, B, \pi)$, the parameters can be trained by first estimating the emission matrix B, which take the form of a Gaussian mixture. The transition matrix A is then calculated statistically according to the state indices of feature vectors in the training set. The initial state distribution is usually assumed to be uniformly distributed [24].

4.3.1.4 Spike-based Classification

The classification schemes discussed so far work well for "traditional" feature representations. Recent developments in spike-based signal processing have led to research in classification methods suitable for spike trains. For example, the *liquid state machine* is a computational model for analyzing spike trains [67]. A related development is known as the *echo state network* [50].

CHAPTER V

ANALOG SOUND CLASSIFIERS

In modern electronic systems, neither analog nor digital signal processing can exist in current technologies without the other. Real-world signals are analog in nature while much of modern control and communication is digital. Traditionally, analog circuits are primarily used for front-end processing, while the core of the computation has been an exclusive feature of digital processors. Advances in analog VLSI circuits, however, have made it possible to perform operations using analog circuitry that were typically done in DSP applications.

A new design approach—cooperative analog/digital signal processing (CADSP)—was recently proposed to combine analog signal processing and digital signal processing techniques for real-world signal processing applications [43]. In this paradigm, part of the signal processing is done in the analog domain before the analog-to-digital converter (ADC). The comparison between traditional signal processing and CADSP is depicted in Figure 17. By using power-efficient analog signal processing circuits, we can reduce the amount of computation done in the DSP, and hence reduce overall power consumption. In this chapter, we review some of the analog design techniques used in the front-end of sound classification systems.

5.1 Analog Audio Feature Extraction

Since the advent of digital signal processing in the 1970s, sound processing has largely been done in the digital domain. Such applications as speech/audio coding, sound enhancement, and synthesis can be efficiently implemented in DSPs or digital application-specific integrated circuits (ASICs). Today, almost all consumer-oriented audio devices are dominated by digital technologies, with two exceptions—high-end audio products and ultra-low power applications. The first category is geared toward audiophiles, who often prefer the "tube sound" of amplifiers based on vacuum tubes. The latter are portable and embedded systems that take advantage of power-efficient analog signal processing circuits. The renewed

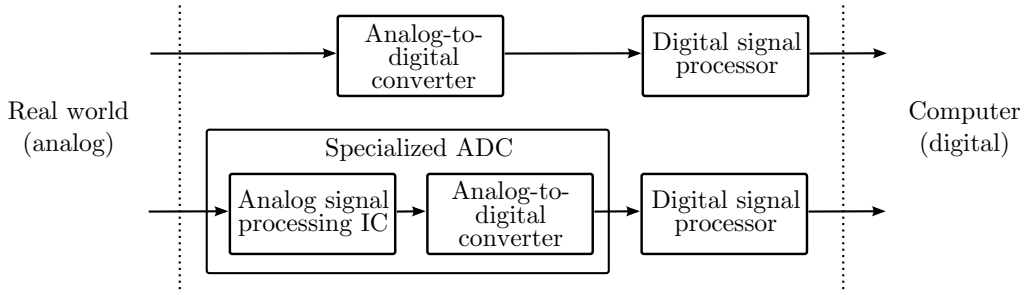


Figure 17: Comparison of traditional signal processing paradigm and CADSP [43]. Typically, real-world signals coming from the sensor are analog in nature. One approach is to put an ADC as close to the sensor signals as possible and allow the remainder of the computations to be performed digitally. The CADSP approach is to perform some of the computations using analog signal processing, requiring simpler ADC and reducing the computational workload of the subsequent digital processing.

interest in analog signal processing was in part sparked by the effort to borrow information processing strategies in biological system by Carver Mead and his colleagues since the late '80s [68]. He coined the term "neuromorphic" to describe analog VLSI systems that mimic neural and biological architectures.

One of the early neuromorphic systems was an analog electronic cochlea developed by Lyon and Mead [66]. Built in CMOS VLSI, the system models the cochlea and higher levels of the auditory nervous system. The model incorporates an asymmetric low-pass/band-pass response at each output channel. In particular, the electronic cochlea features a cascade of simple, near-linear, second-order filters with controllable Q (quality factor) parameters sufficient to capture the physics of the fluid dynamic traveling-wave system in the cochlea. The cascade filter model is implemented using transconductance-capacitance (G_m-C) circuits, as illustrated in Figure 18. Watts et al. improved the original cochlea circuit by increasing the input dynamic range of the operational transconductance amplifiers (OTA) and eliminating a large-signal instability [108]. The improved implementation is shown in Figure 19.

Many versions of "silicon cochleas"[1] have since been created, each generation improving upon previous designs. For example, van Schaik et al. implemented the cochlea cascade

[1]Although there are many semiconductor materials, silicon is by far the most prevalently used material in integrated circuits. Neuromorphic systems are often implemented on silicon CMOS technologies because the processes are relatively affordable; it also makes integration with interfacing circuitries—which are almost always fabricated on CMOS—easy.

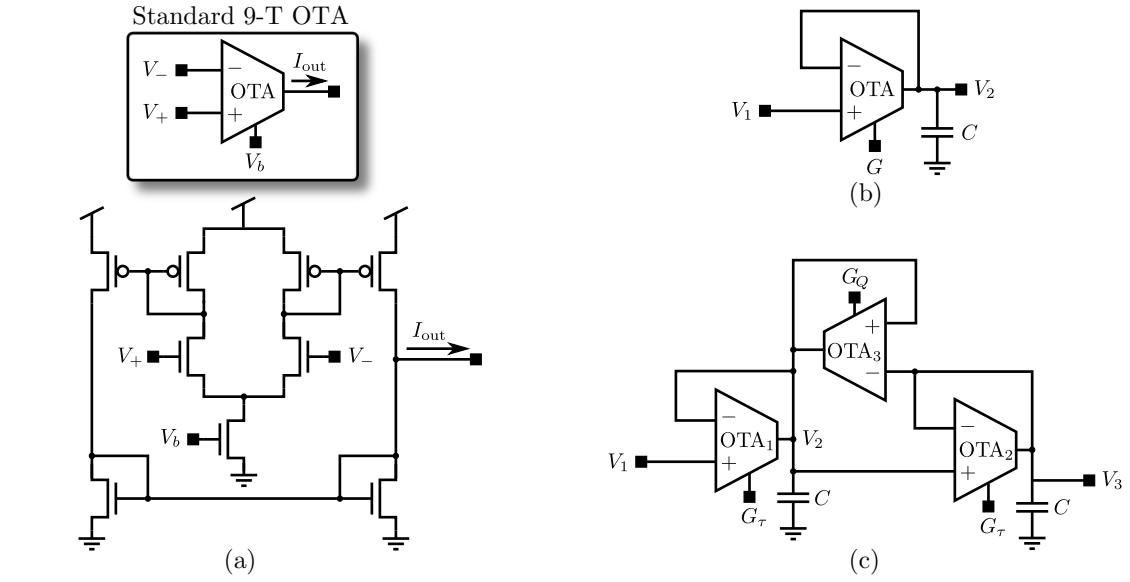


Figure 18: Second-order filter sections used in Lyon and Mead's analog electronic cochlea [66]. (a) A nine-transistor (9-T) implementation of the operational transconductance amplifier (OTA) is used in the G_m-C filters. The active devices operate in *subthreshold* (weak inversion) region. (b) The first-order follower-integrator circuit has a transfer function $\frac{V_2}{V_1} = \frac{1}{\tau s+1}$, where $\tau = {}^{C}/_{G}$, and G is the transconductance of the OTA. (c) The second order filter can be thought of as a cascade of two feed-forward follower-integrator circuits and a feedback OTA that amplifies the difference between feed-forward circuits. If OTA_3 has zero gain, then the transfer function is simply $\frac{V_3}{V_1} = ({}^{1}/_{\tau s+1})^2$, where $\tau = {}^{C}/_{G_\tau}$. However, OTA_3 introduces resonance by providing positive feedback, and its gain controls the Q factor. The overall response is $\frac{V_3}{V_1} = \frac{1}{\tau^2 s^2+2\tau s(1-\alpha)+1}$, where $\tau = {}^{C}/_{G_\tau}$ and $\alpha = {}^{G_Q}/_{2G_\tau}$.

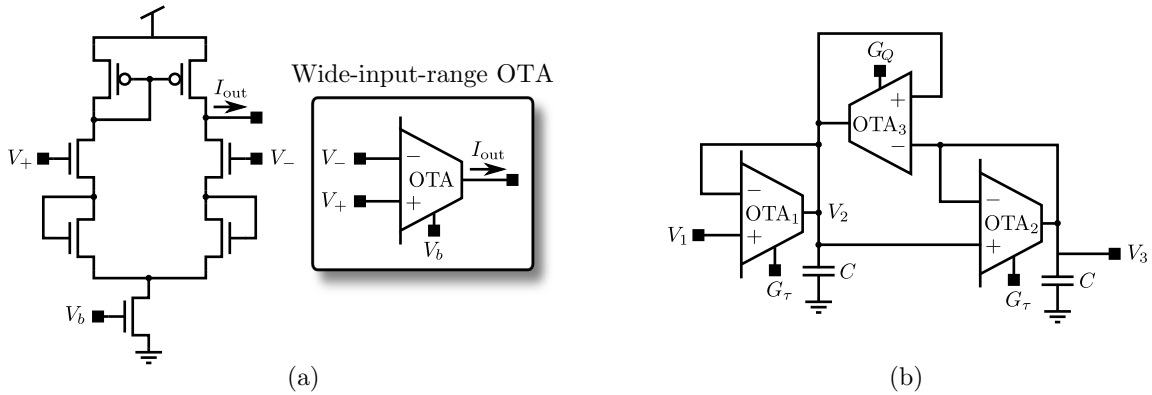


Figure 19: The improved second-order filters introduced by Watts et al. [108]. (a) The linear range of the subtreshold OTA is increased by using source degeneration. The circuit shown here is the "basic" version; a wide-output-range version similar to the one in Figure 18(a) can also be used. (b) Large-signal instability is eliminated by using wide-input-range OTA in the feed-forward path and a standard OTA in the feedback path.

with compatible lateral bipolar transistors (CLBTs) [104]. This scheme reduces the bias transistor variations, a major source of mismatches of the cochlea's parameters. A low-power wide-dynamic-range analog VLSI cochlea was presented by Sarpeshkar et al [91]. The linear range of the OTAs is enhanced by the novel techniques of gate degeneration and bump linearization.[2] This design also features dynamic gain control (AGC), which detects and compresses high-level input signals. Some of the electronic cochleas were developed to be used in cochlear implants, but the filter cascade provides an excellent frequency decomposition scheme in analog audio feature extraction.

Sub-band processing requires an efficient filter bank to separate the input signal into multiple components. Therefore, band-pass filters play an important role in audio feature extraction. Salthouse and Sarpeshkar presented a programmable band-pass filter consuming micro-watts of power for use in bionic ears [91]. The filter uses capacitive-attenuation to increase input linear range and incorporates an offset adaption mechanism. Graham et al. demonstrated a power-efficient programmable band-pass filter suitable for large filter-bank applications [38]. The filter section is based on a capacitively coupled current conveyor (C^4) design (see, for example, [99]). Programmability of the time constants can be achieved by using floating-gate transistors as current sources. A band-pass filter with inherent gain adaptation was introduced by Odame et al. [73]. Its intended use is in hearing aids and cochlear implants, which require input dynamic range compression to compensate for the lost ability of a damaged cochlear to adapt to the signal level. In this circuit, automatic gain control is made possible by using a nonlinear transconductance element, whose Q factor decreases as the input amplitude increases, hence dampening the input signal.

Another important component of electronic cochleas is the inner hair cell model. At a basic level, inner hair cells convert sound to nerve signals. The transduction is modeled by rectification and differentiation and is often followed by a conversion from voltage to current. In addition, the overall dynamic range of the system can be improved by adjusting the filter

[2] In the wide-linear-range OTA, the *well terminals* of the input differential-pair transisotrs are used as the amplifier inputs. A bump linearization technique, which essentially removes cubic distortion from subthreshold differential pairs, further extends the linear range. This design, however, increases the second-order distortion. The amplifier circuit is detailed in [90].

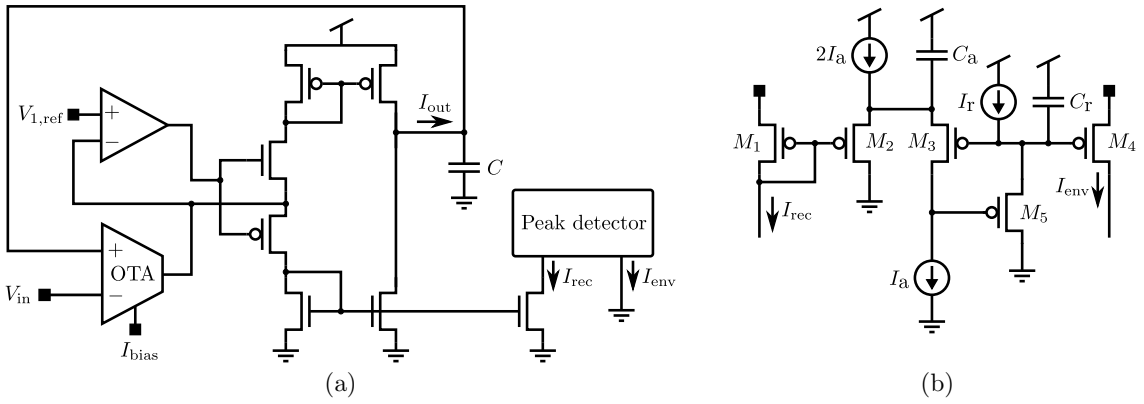


Figure 20: The wide-dynamic-range envelope detector presented by Zhak, Baker, and Sarpeshkar [113]. (a) The class-B current mirror consists of a classical current conveyor (see, for example, [96]) and a feedback circuit that minimizes the "dead zone." Note that the schematic in the reference has a sign error in the feedback amplifier. (b) The peak detector consists of a log-domain, current-mode filter (see, for example, [81]) operating in the subthreshold domain. The source voltage of transistor M_2 is proportional to the logarithm of the low-pass filtered input current. During the *attack* phase, M_3 acts as a level shifter, and the time constant is given by $\tau_{\mathrm{a}} = {}^{C_{\mathrm{a}} V_{\mathrm{T}}}/_{\kappa I_{\mathrm{a}}}$, where V_{T} is the thermal voltage and κ is the capacitive divider ratio between the gate and the channel. The M_3-M_5 feedback loop helps fix the currents by clamping the gate voltage of M_5. During the *release* phase, M_5 is off, and the time constant is given by $\tau_{\mathrm{a}} = {}^{C_{\mathrm{r}} V_{\mathrm{T}}}/_{\kappa I_{\mathrm{r}}}$.

response. This can be accomplished by detecting the envelope of the inner hair cell output and modulate the time constant and Q factor of the filter bank accordingly. Sarpeshkar et al. took this approach in their wide-dynamic-range cochlea model [91]. Envelope detection turns out to be a useful operation in estimating the energy of an audio signal. Baker, Zhak, and Sarpeshkar later refined the envelope detector [6, 113]; the circuit is depicted in Figure 20. A similar implementation is used in a glass break sound detector designed by Gestner et al. [37]. Yet another simpler design, as illustrated in Figure 21, was presented by Twigg [102]. This design has been programmed and validated in the field programmable analog array (FPAA), an analog counterpart of the ubiquitous field programmable gate array (FPGA) [9]. Smith also presented a number of peak detector architectures to be used in an analog auditory feature extraction and recognition system, which is to be discussed later in this section [98]. In particular, a floating-gate-based peak detector produces the logarithm of the amplitude at its output.

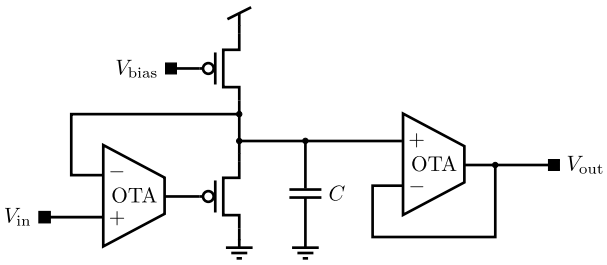


Figure 21: The envelope detector discussed in Twigg's dissertation [102]. If the input signal is decreasing, the bottom transistor quickly discharges the capacitor, so the output follows the input. On the other hand, if the input signal is increasing, the bottom transistor is turned off, and the top transistor charges the capacitor. Therefore, the circuit detects the *minimum* of the input signal. The (release) time constant is controlled by the charging current set by V_{bias}. The current source can also be replaced by a programmable floating-gate device.

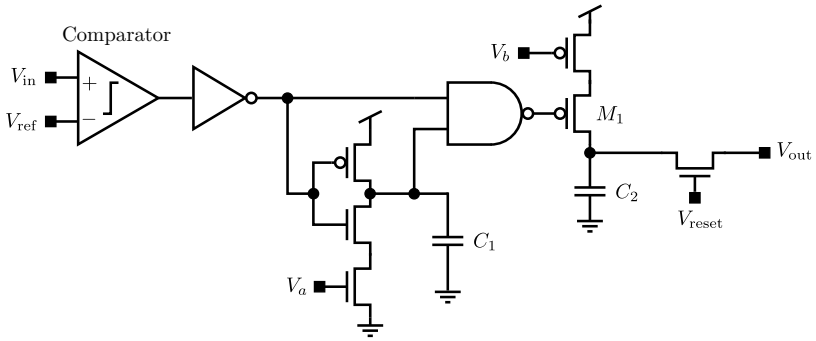


Figure 22: The zero-crossing circuit described by Gestner et al. for use in a glass break sound detector [37]. For each negative-to-positive signal transition, the NAND gate turns on M_1 momentarily, which charges C_2. The length of the pulse is controlled by V_a, and amount of charge dumped on C_2 is controlled by V_b. The voltage across C_2 increases linearly with respect to the number of zero crossings. At the end of the pre-determined interval, C_2 is reset by V_{reset}.

As discussed in Section 4.1.2, zero-crossing rate is often utilized in audio classification. It is also used as a signal representation in neural models of auditory processing in the inner ear hair cells. Inspired by this model, Kumar et al. designed an analog VLSI chip for auditory feature extraction using level crossing [57]. The system uses a level crossing circuit they presented earlier [56]. The schematic is reproduced and discussed in Figure 23. Moreover, Gestner et al. included zero-crossing rate as a feature in their analog glass break detector [37]; the circuit is shown in Figure 22 and described in detail in Section 6.4. These circuits compute related but different quantities: The first measures the time interval

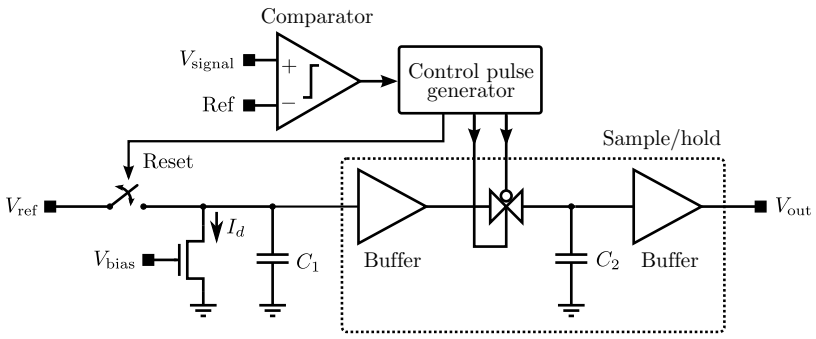


Figure 23: The level crossing time interval circuit presented by Kumar, Cauwenberghs, and Andreous [56]. A constant current I_d is integrated onto capacitor C_1 over the time interval between consecutive upward level crossings. The capacitor voltage is sampled at the end of the interval and, simultaneously, reset for integration for the next cycle. At any time, V_{out} measures the most recent level crossing interval. The control pulse generator ensures that the capacitor voltage is reset *after* the result is sampled and held.

between subsequent level crossings, while the latter estimates the number of level crossings per unit time.

Hasler, Smith, and others proposed an analog audio processing front-end [46, 100]; this work is detailed in Smith's dissertation [98]. The analog auditory sensing system interfaces they developed are an example of CADSP (Figure 17). Several analog signal processing building blocks are presented: the C^4 second-order filters, peak detector circuit, and multiplier. Using these building blocks, the authors built two analog front-ends for audio signal processing systems. The first system is a noise suppression front-end for speech enhancement. As shown in Figure 24, the system uses dynamic sub-band gain control to reduce overall noise in noisy signals. The second system is an analog front-end for speech recognition, which implements an analog version of MFCC (Section 4.1.4). Figure 25 depicts a simplified block diagram of the sensing interface.

Other analog features extraction circuits have been reported in the literature. For example, Fragnière et al. proposed a a system to extract LPC (Section 4.1.5) using a cochlear model [88]. Chakrabartty and Gore also introduced a sigma-delta analog-to-LPC feature converter for portable recognition interfaces [19]. Recently, Chan et al. fabricated a matched silicon cochlea pair with address event representation (AER) interface [20]. The system enables spike-based processing for such applications as sound localization and visual-audio

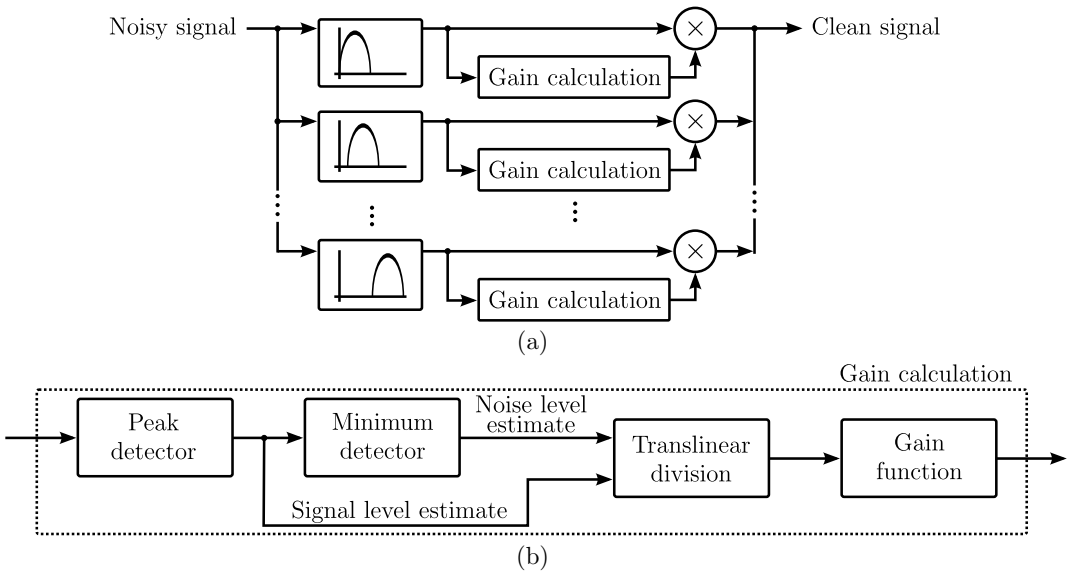


Figure 24: Analog noise suppression front-end for speech enhancement proposed by Hasler et al. [46]. (a)The incoming noisy signal is divided into exponentially-spaced frequency bands. The gain for each band is then computed and applied to the sub-bands. (b) The gain calculation block estimates the noise level and noisy signal level. The ratio of these two parameters becomes the gain. Here, division is done in current-mode translinear circuits.

sensor fusion. Another spike-based feature extraction system is the speech detection wake-up circuit demonstrated by Delbrück et al. [28].

5.2 Analog Signal Classification

The classification algorithms discussed in Section 4.2 are often simulated in computers and implemented on embedded digital platforms, but they have also been built in analog systems. In this section, we review the analog realizations of the classification stage.

The analog implementation of the artificial neural networks (see Section 4.2.4), which is a mathematical representation of biological neural networks, are well reported in the literature. Hasler and Akers presented an analog, multiple-input, multiple-output neural network [41]. The system features multilevel analog dynamic random access memories (DRAMs) to store coefficient weights. The authors also presented an efficient hardware implementation of a training algorithm, the generalized Hebian algorithm [42]. Lont and Guggenbühl demonstrated a CMOS implementation of a multilayer analog perceptron with nonlinear synapses [64]. Montalvo et al. proposed a general-purpose VLSI neural network

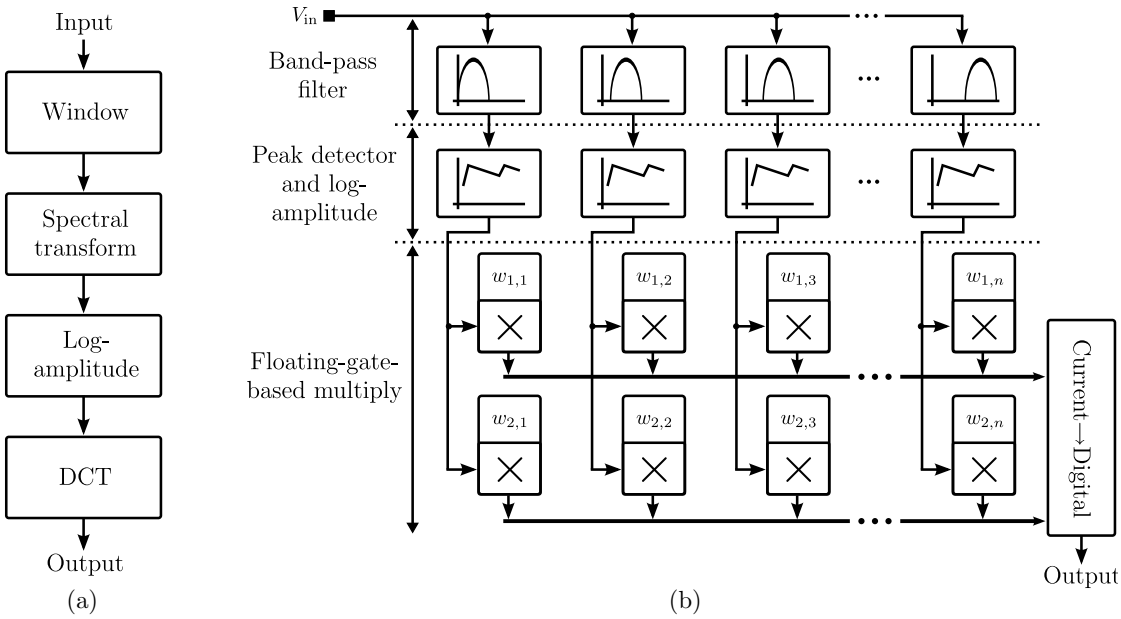


Figure 25: Comparison of conventional and analog cepstrums, as presented by Hasler, Smith, and others [46, 98]. (a) The block diagram depicts the processing step to compute the cepstrum (discussed in Section 4.1.4). (b) The simplified block diagram shows the computation of continuous-time cepstrum, which approximates mel-cepstrum or cepstrum. The system contains a bank of band-pass filters, peak detectors that measure the log-amplitude of the sub-band signal, and floating-gate based circuitries that implements matrix multiplication. The last component effectively computes the DCT.

with on-chip learning [70]. The chain rule perturbation (CHRP) algorithm, a learning method tolerant of analog nonidealities, is used in the design. Milev and Hristov reported an analog implementation of neural network utilizing the circuit's inherent quadratic nonlinearity [69]. Recently, Gatet et al. presented a CMOS neural network integrated in an optoelectronic measurement system, which enables real-time surface classification applications [35].

Chakrabartty and Cauwengerghs derived a margin propagation algorithm for decoding [17]. The algorithm employs the concept of reverse water-filling normalization, and it depends on basic additions and subtractions, making the architecture suitable for analog VLSI implementation. The authors designed an analog pattern classifier based on this algorithm [18]. The proposed system adopts a support vector machine classifier (discussed in Section 4.2.5) using a quadratic kernel, but the algorithm can be used to design other classifiers. An analog support vector machine based on Gaussian kernels was presented by Kang and Shibata [53].

Other classification algorithms have been built in analog systems. Gestner et al. implemented a linear discriminant function of the form $\sum_{i=1}^{N} a_i x_i \geq b$ (discussed in Section 4.2.3); the classifier is trained by the Ho-Kashyap procedure [37]. Peng et al. presented an analog classifier based on multidimensional radial basis functions (discussed in Section 4.3.1) [79]. Their system utilizes a programmable floating-gate bump circuit [80]. Analog classifiers inspired by the hidden Markov model (also discussed in Section 4.3.1) have also been reported in the literature [44, 45]. As the spike-based signal processing paradigm becomes more popular, so does spike-based classification. For example, Oster et al. formalized the use of winner-take-all circuits in decision making, taking into account the input statistics and neuron response variance [76].

CHAPTER VI

SOUND DETECTOR FRONT-END ON A PROGRAMMABLE ANALOG PLATFORM

Intelligent audio sensors are devices or subsystems that actively monitor the environment using acoustic information. They are often embedded in larger systems to enhance functionality or aid decision making. For example, speech signal detectors are used to activate mobile devices or help suppress background noise [105]. Gunshot detectors are used to identify and locate gunfire events for security and military applications [11]. The implementation of sound detectors often follows the traditional digital signal processing paradigm, in which the acoustic signal is digitized in the front end of the system, and the processing is performed in the digital domain. In this case, the system is designed for a digital signal processor (DSP), a field programmable gate array (FPGA), or a digital application-specific integrated circuit (ASIC).

The purpose of the audio sensor is to react to the occurrence of certain sounds, which can be an infrequent event (e.g., speaking to the phone or gunshot). However, the analog-to-digital converter (ADC) and digital circuitry must remain active all the time for continuous detection. This presents a problem for portable and embedded devices, where power consumption needs to be minimized. One approach to mitigate this problem is to operate the digital system with a small duty cycle. In this chapter, we present an alternative solution—using a low-power analog front-end to wake up a more powerful digital system.

Analog signal processing systems have been shown to require less power than their digital counterparts if the desired signal-to-noise ratio is below a certain value [89]. Advances in floating-gate-transistor programing techniques have enabled precise programming of large arrays of analog elements [38]. In fact, using floating-gates, the researchers have proven that the analog approach to design sensing system interface is practical [46]. Recent developments in field programmable analog arrays (FPAA), the analog counterpart of FPGA,

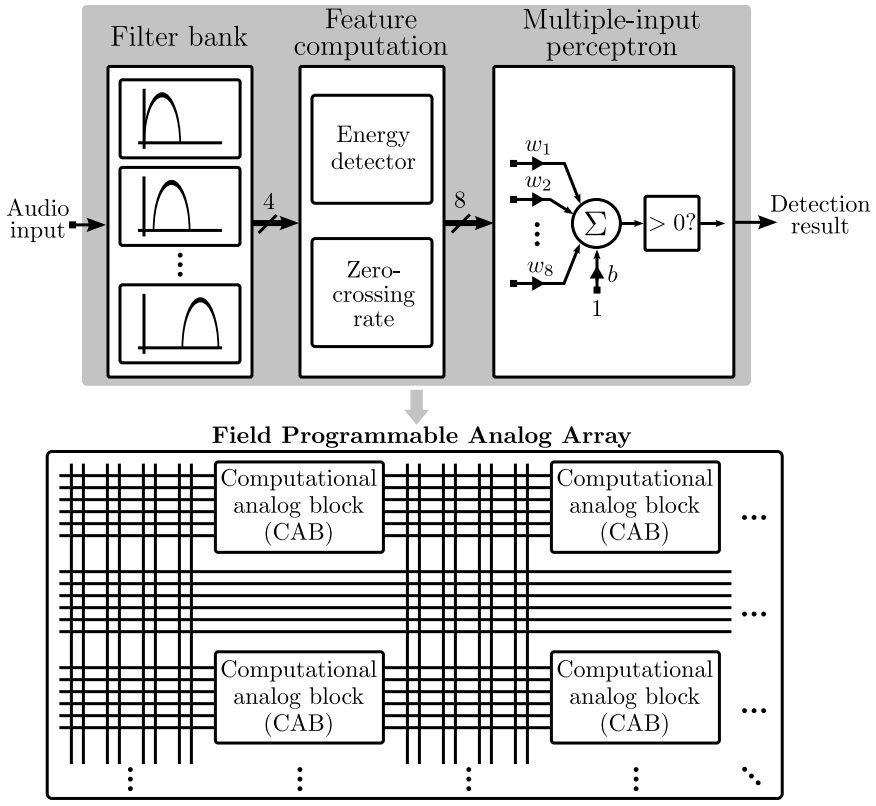


Figure 26: Overview of the sound detector front-end implemented on programmable analog hardware. In the *feature extraction* stage, the input audio signal is decomposed into several frequency bands, and the instantaneous energy and zero-crossing rate of the sub-band signal are estimated. In the *classification* stage, the audio features are classified by a single-layer perceptron into two categories of sound. The neural network is trained off-line. The system is implemented on a floating-gate-based field programmable analog array (FPAA) [8]. The FPAA comprises 32 computational analog blocks (CAB), which are connected by a programmable switching fabric. In addition, the floating-gate-based switch element in the switching fabric can be used for computational purpose.

have allowed us to design and prototype the analog front-end of a sound detector. The architecture of the proposed system and our design approach are depicted in Figure 26.

6.1 Analog Processing for Sound Detection

6.1.1 Cooperative Analog-Digital Signal Processing

In modern sensing systems, neither analog nor digital circuits can exist in current technologies without the other. Advances in analog VLSI circuits, however, have made it possible to perform operations that were typically done in the digital domain. The cooperative analog/digital signal processing (CADSP) design approach was proposed to combine analog

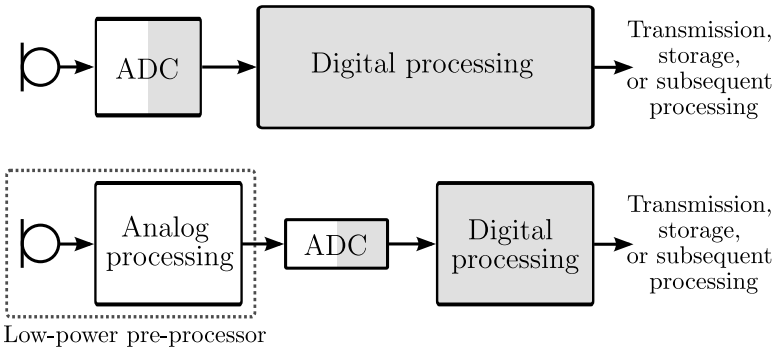


Figure 27: CADSP approach to design sensing systems. Real-world signals received by the sensor are typically analog in nature. In the traditional design paradigm, the ADC is placed as close to the sensor as possible, and the remainder of the computation is done digitally. The CADSP approach is to perform part of the computation using analog signal processing, requiring a simpler ADC. Here, the block size represents typical design complexity and power consumption. The shaded areas denote digital signal representation.

and digital signal processing techniques for real-world signal processing applications [43]. The CADSP design paradigm is illustrated in Figure 27 in the context of an intelligent audio sensor. By using power-efficient analog signal processing circuits, we can reduce the amount of computation done in the digital domain, reducing overall power consumption.

6.1.2 Feature Extraction

In the feature extraction stage, we characterize and summarize the incoming audio signal. The most widely used audio features were discussed in Section 4.1. Some of these audio features are especially suitable for analog implementation. Delbrück et al. employed energy estimation in phoneme bands in a speech signal detector [28]. Gestner et al. used full-band energy, high-passed energy, and zero-crossing rate as features to detect glass-break sound [37]. These features are extended to include *sub-band energy estimation* and *sub-band zero-crossing rates*. In this chapter, we present the hardware results for these two sets of audio features.

Sub-band processing mimics the auditory processing in biological systems. In human auditory systems, the basilar membrane can be modeled as a bank of band-pass auditory filters. Although sound discrimination in human is a complex process, the energy distribution across the *critical bands* is an important cue in sound discrimination. Specifically, the band-passed energy of a signal is a representation of the signal that balances between time

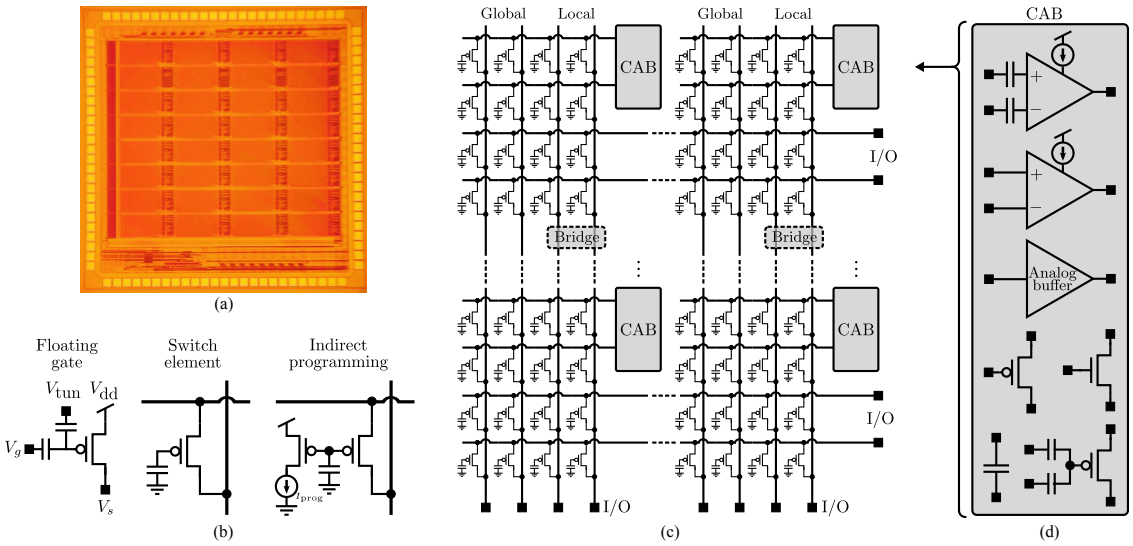


Figure 28: Reconfigurable Analog Signal Processor (RASP). RASP 2.8a is a programmable analog processor in the FPAA family [8]. (a) The die photo of RASP 2.8a is shown here. The chip is fabricated in a 0.35-µm CMOS technology. (b) The floating-gate transistor is used as both a digital switch and a programmable analog storage element. (c) RASP 2.8a consists of 32 computational analog blocks (CABs) that are connected by programmable switching fabric. (d) A wide range of analog computational elements are available in the CABs.

and frequency resolution [85]. On the other hand, the zero-crossing rate of a filtered signal represents the *dominant frequency* of the signal and is widely used for signal discrimination [54].

6.1.3 Analog Detection and Classification

In the classification stage, we assign the signal into different categories based on the extracted features. An important aspect of classification is to obtain a model from the training data—this is the process of learning. Based on the model, the classification stage predicts something about data that it has not encountered before. As reviewed in Section 4.2, an important assumption is that the test data comes from the same probability distribution as the training data. Specifically, we can consider independent and identically distributed (i.i.d.) data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \in \mathcal{X} \subset \mathbb{R}^D$ is a D-dimensional vector and y_i is in some finite set $\mathcal{Y}$. A *classification rule* is a function $h : \mathcal{X} \to \mathcal{Y}$. When a new x is observed, y is predicted to be $h(x)$. Here, we focus on the special case in which the classification system is a two-class detector, i.e., $\mathcal{Y} = \{0, 1\}$.

Supervised learning algorithms are broadly categorized into two approaches—*generative* and *discriminative*. In the first approach, one models the distribution of the input data; however, this approach can be computationally demanding and requires a large amount of training data. In the second approach, one directly estimate the decision boundary between the classes. In the two-class case, where $Y \in \mathcal{Y} = \{1, -1\}$, the classification rule is given by

$$h(x) = \begin{cases} 1, & \text{if } r(x) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{55}$$

Here, the equation $r(x) = 0$ describes the decision boundary. A simple model for r is the *perceptron*, a generalized linear model. In this model, the separating plane in the feature space is given by the equation

$$r(x) = \beta_0 + \sum_{i=0}^{N} \beta_i x_i = 0, \tag{56}$$

where N is the number of features. Because of its simplicity, the perceptron has been implemented in the analog domain. For example, Gestner et al. implemented a three-input analog perceptron to detect glass-break sound, training the classifier off-line using the Ho-Kashyap procedure [37]. The perceptron is a special case of the artificial neural network (ANN), a mathematical approximation of biological neural networks. The analog implementation of ANNs is well reported in the literature, as reviewed in Section 5.2. For example, Hasler and Akers presented an analog, multiple-input, multiple-output neural network [41]. The system features multilevel analog dynamic random access memories (DRAMs) to store coefficient weights. More recently, Lont and Guggenbühl demonstrated a CMOS implementation of a multilayer analog perceptron with nonlinear synapses [64].

In this chapter, we present a multiple-input perceptron (single-layer ANN) that is based on subthreshold, current-mode multiplication. Our design is enabled by recent developments in efficient and precise floating-gate programming, which is crucial in storing the classifier weights. A current-mode floating-gate-based ANN was introduced by Borgstrom et al., but the system lacked reconfigurability and scalability [13]. Also, they designed the transistors to run above threshold, which would consume higher power. More recently, Zatorre et al. presented an analog processor for signal conditioning based on multiple-layer

perceptron [112]. However, their design relied on digitally programmable circuits to store analog weights.

6.2 Programmable Analog Platform

FPGAs are a standard prototyping and implementation platform for digital circuits. The analog counterpart of FPGA is the FPAA. Traditionally, analog circuit design follows a design-simulation-fabrication-testing-redesign cycle, usually requiring months for each design iteration, but recent developments in FPAAs have brought the level of ease and flexibility of digital circuit design to the analog domain. Modern FPAAs are mixed-signal systems that provide both a rapid prototyping and a final implementation platform for analog, mixed-signal, and neuromorphic circuits. The users of the device can design, program, and test large scale systems in a matter of minutes, saving considerable development time and fabrication costs. The analog front-end for sound detection presented in this chapter is implemented on RASP 2.8a, an FPAA designed for building generic analog computational systems [8, 9]. A die photo of RASP 2.8a is shown in Figure 28(a). The chip is fabricated in a 0.35-µm CMOS technology, with a 2.4-V V_{dd}.

The essential circuit element that facilitates highly dense FPAAs is the floating-gate transistor, which has an electrically isolated gate that allows the device to store and retain charge (Figure 28(b)). In the swtiching fabric this element is used as a routing switch to connect crossbars in the switching fabric and a programmable voltage/current reference; it can also be used as a computational element to build more complex structures [103]. The architecture of RASP 2.8a is shown in Figure 28(c). There are 32 CABs embedded in the programmable routing fabric. Both global and local level interconnections are available. Global routing lines are used to connect CABs that are far apart on the chip, and local routing lines connect elements in the same CAB or nearby CABs. Global routing lines also connect to input/output ports to interface with off-chip components. As illustrated in Figure 28(d), the CABs contain components commonly used in analog computation: OTAs with a programmable bias, OTAs with a programmable input and bias,

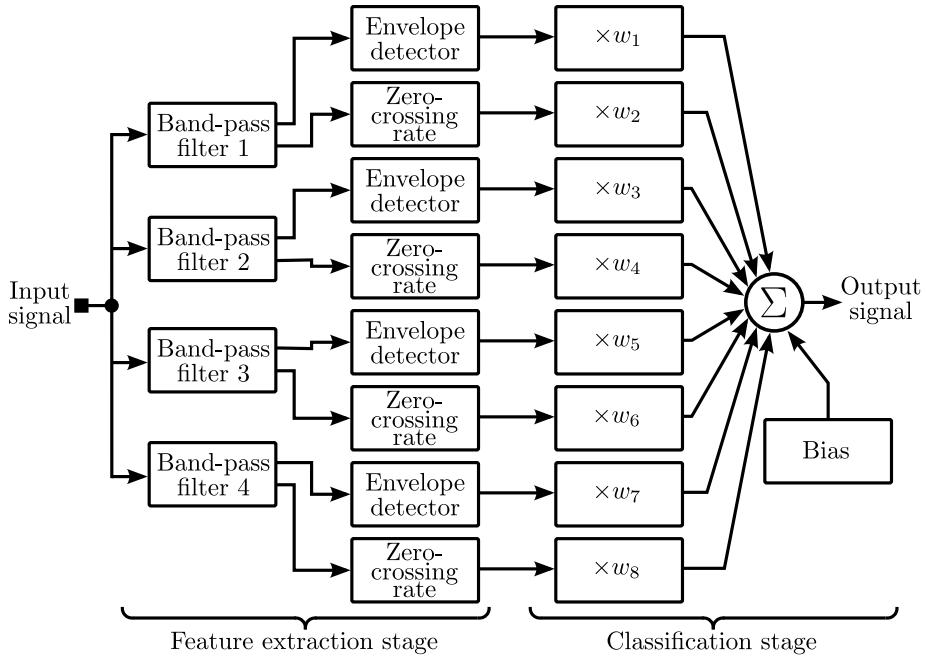


Figure 29: Block diagram of the analog front-end. In the feature extraction stage, the signal is spectrally decomposed into four sub-bands, and the envelope and zero-crossing rate of the sub-band signals are computed. In the classification stage, the eight features are weighted and added to a bias. This multiple-input perceptron computes the detection output.

analog buffers, NFETs/PFETs, capacitors, transmission gates, and multiple-input translinear elements (MITE).

6.3 System Components

The analog front-end is divided into two stages. The feature extraction stage is used to characterize the input signal; and the classification stage is used to decide which category the signal belongs to. The block diagram of the entire system is illustrated in Figure 29. The circuits to implement the two stages are discussed below.

6.3.1 Feature Extraction

In the feature extraction stage, a band-pass filter bank performs frequency decomposition on the input audio signal. We then estimate the zero-crossing rate and energy of the sub-band signal, as explained in Section 6.1.

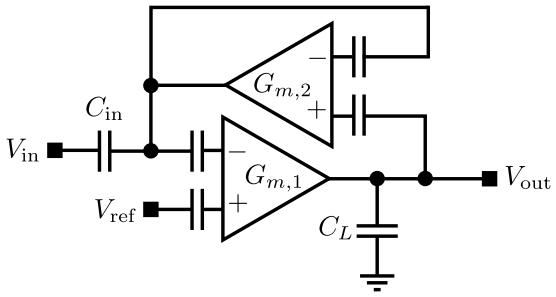


Figure 30: Schematic of the band-pass filter implemented on RASP 2.8a. The design of this second-order section is inspired by that of the capacitively coupled current conveyor (C^4) band-pass filter [38]. We use OTAs (instead of transistors) to implement the transconductance elements because they are plentiful on the FPAA. Floating-gate-input OTAs are employed to minimize input offsets.

6.3.1.1 Filter Bank

We use transconductance-capacitance (G_m-C) filters to implement the band-pass filter bank. The schematic of the second-order section is shown in Figure 31. The filter design is inspired by that of the capacitively coupled current conveyor (C^4) band-pass filter [38]. However, instead of integrated transistors, we use operational transconductance amplifiers (OTAs) to implement the transconductance elements for two reasons. First, the OTAs are standard analog computational elements on RASP 2.8a, which contains a large number of OTAs. Second, the bias current I_b of the OTAs can be programmed using on-chip floating-gate transistors. If the OTA is operating in the subthreshold region (when the transistor drain currents are less than $I_{\text{threshold}}$, about 100 nA in the current process), then the transconductance is given by

$$G_m = \alpha I_b \frac{\kappa}{2U_T}, \tag{57}$$

where α is an attenuation factor due to the input capacitive divider, κ is the capacitive coupling ratio of the input differential pair (including the input capacitive divider and gate-to-channel coupling), and U_T is the thermal voltage. By controlling I_b, we can tune the G_m and hence the frequency response of the filter. OTAs with differential floating-gate inputs are used to minimize input offset. They also provide a larger input linear range than OTAs with non-floating-gate inputs do.

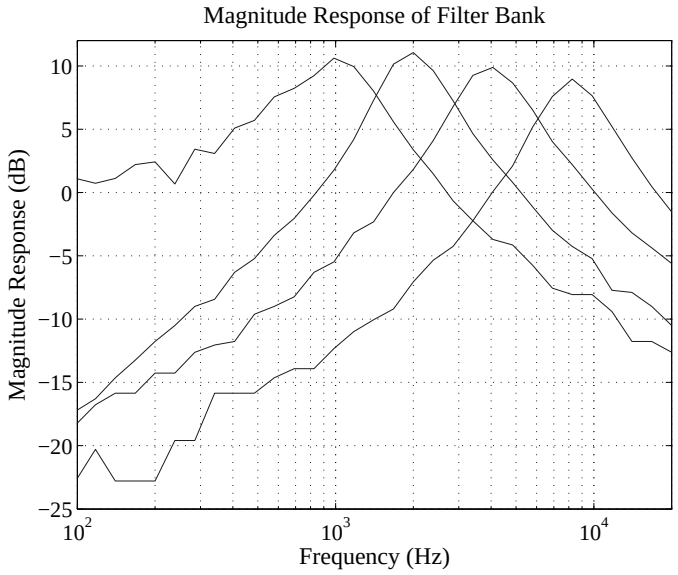


Figure 31: Measured magnitude response of the filter bank on RASP 2.8a. Four band-pass filters, whose center frequencies are exponentially centered, are used for frequency decomposition. The roughly 20-dB-per-decade roll-off is shown to provide sufficient spectral separation while covering the audible range.

The transfer function of the filter is

$$\frac{V_{\text{out}}}{V_{\text{in}}} = -\frac{s\frac{G_{m,1}}{C_L}}{s^2 + s\frac{G_{m,2}}{C_{\text{in}}} + \frac{G_{m,1}G_{m,2}}{C_{\text{in}}C_L}}, \tag{58}$$

where $G_{m,1}$ and $G_{m,2}$ are the transconductance of the OTAs, and the other parameters are defined in the schematic. The center frequency of the band-pass filter is

$$f_c = \frac{1}{2\pi}\sqrt{\frac{G_{m,1}G_{m,2}}{C_{\text{in}}C_L}}, \tag{59}$$

and the quality factor is given by

$$Q = \sqrt{\frac{G_{m,1}}{G_{m,2}}\frac{C_{\text{in}}}{C_L}}. \tag{60}$$

A bank of four band-pass filters is programmed to have constant Q and exponentially spaced f_c; the magnitude response of the filter bank is shown in Figure 31. On RASP 2.8a, it is easy to include more filters in the filter bank, but doing so would increase power consumption. The center frequencies and quality factor of the four filters are chosen to cover most of the audible range (50 Hz to 15 kHz) while providing sufficient separation from adjacent channels.

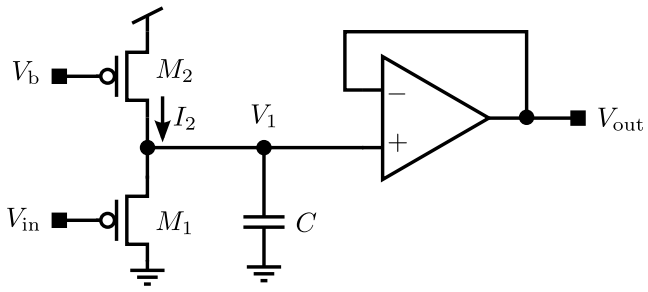


Figure 32: Schematic of the envelope detection circuit implemented on RASP 2.8a. If the input V_{in} is decreasing, the transistor M_1 acts as a source follower such that V_1 quickly follows V_{in}. However, when V_{in} is increasing, it charges the capacitor C, and V_1 increases at a rate of I_2/C, where I_2 is the drain current of M_2. Therefore, the circuit detects the minimum envelope of a signal (cf. Figure 21).

6.3.1.2 Envelope Detector

We use an envelope detector to estimate the energy of a sub-band signal. The schematic of the envelope detector is depicted in Figure 32. The nonlinear circuit behaves differently for increasing and decreasing input voltage, V_{in}. We assume that the transistors are in subthreshold. When V_{in} is decreasing and $V_{in} < V_1$, the lower transistor M_1 is a source-follower, and V_1 follows V_{in} with a gain of approximately κ (capacitive coupling ratio); When V_{in} is increasing and $V_1 < V_{in}$, however, V_1 cannot follow V_{in} immediately because M_1 is at cut-off. In this case, the current of M_2 charges up the capacitor C at a rate of I_2/C, where I_2 is the drain current of M_2. Since the circuit quickly follows a falling V_{in} and slowly rises following a rising V_{in}, it detects the minimum envelope of the input signal. The OTA-based buffer isolates V_1 from the output voltage V_{out}. The buffer makes the time constant of the envelope detector independent of routing and prevents loading. The transfer function of the circuit is shown in Figure 33. A detailed discussion of the dynamic behavior of this circuit can be found in [84].

We can change the bias current I_2 to tune the "attack" and "release" time of the envelope detector. The detector response must be sufficiently slow relative to the signal itself so that it tracks the modulating information while ignoring the "carrier." On the other hand, the envelope must be fast enough to capture significant changes in the modulating signal. Therefore, it is crucial to select the "attack" and "release" time appropriately. For each

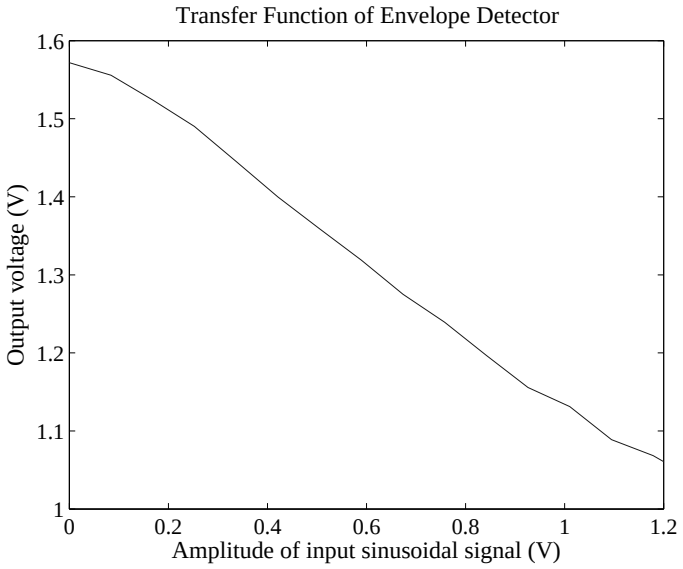


Figure 33: Measured transfer function of the envelope detector on RASP 2.8a. As the amplitude of the input signal goes from 0 to 1.2 V, the output goes from approximately 1.6 to 1 V. Therefore, the circuit tracks the minimum envelope.

sub-band, we set I_2/C such that the charge rate is lower than the highest frequency and higher than the lowest frequency in that frequency band.

6.3.1.3 Zero-crossing Rate

There are two ways to estimate the zero-crossing rate of a signal. One way is to measure the number of zero-crossings per time interval; the other is to measure the time between subsequent zero crossings [56]. We adopt the first approach for the FPAA implementation because it requires fewer circuit components. The circuit is modified from the design in [37].

The schematic of the circuit is shown in Figure 34 (cf. Figure 22). The comparator at the input goes high if the input voltage V_{in} is greater than the reference voltage V_{ref} (the AC zero). When this transition occurs, the one-shot circuit produces a negative pulse, which turns on the charging circuitry and allows the output capacitor C_2 to be charged. An OTA is used to implement the comparator because of its availability on the FPAA; a floating-gate-input OTA is used to minimize the input DC offset. The one-shot circuit works as follows: V_3 is a delayed version of V_1. When V_1 makes a negative-to-positive transition, V_2 and V_3 will be low momentarily, and the pseudo-NAND gate outputs a negative pulse. The

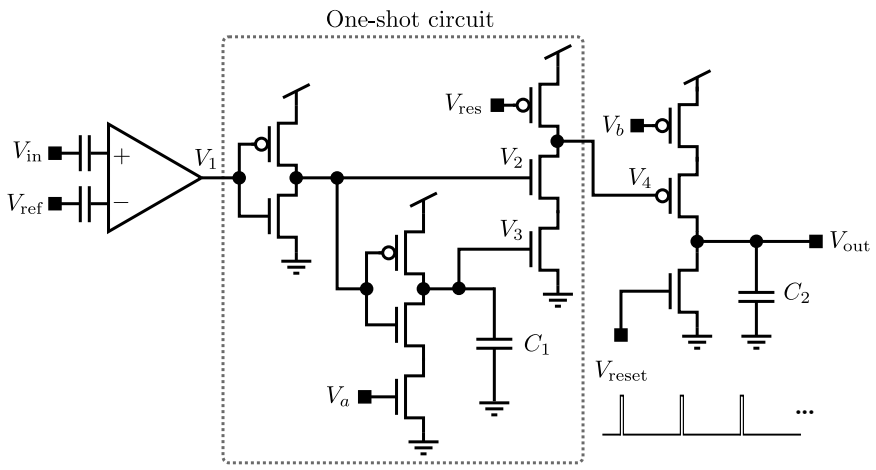


Figure 34: Schematic of the zero-crossing circuit implemented on RASP 2.8a. The input OTA acts as a comparator (using floating-gate inputs to minimize input offset). When the input signal V_{in} makes a negative-to-positive transition (from below V_{ref} to above V_{ref}), the one-shot circuit creates a short pulse, which allows C_2 to be charged up. At the end of a time frame, the V_{reset} signal discharges C_2 and resets V_{out}. This circuit is modified from the one in Figure 22.

width of the pulse is controlled by the leaky (current-starved) inverter input V_a, and the amount of charge transferred to C_2 is controlled by V_b.

The output of the zero-crossing-rate circuit is available at the end of each preset time frame. The output voltage V_{out} is read, and the V_{reset} signal discharges C_2 for the next time frame. The transfer function of the circuit is shown in Figure 35.

6.3.2 Classification

In the classification stage, we implement a multiple-input perceptron. This is reduced to realizing the dot product $r(x) = b + \sum_{i=1}^{8} w_i x_i$, where x_i is the feature, w_i is the weight, and b is the bias. For detection applications, this equation defines the linear surface that separates two classes of signal. We first discuss the vector-matrix multiplier (VMM) circuit, then we describe the VMM-based perceptron.

6.3.2.1 Analog Multiplication and Addition

"Multiply-and-add" (MAC)—essentially a vector-matrix multiplication—is the most important operation on DSPs, but the equivalent operation in analog processors can be 1000 times more power efficient than comparable digital implementations [40]. The FPAA is especially

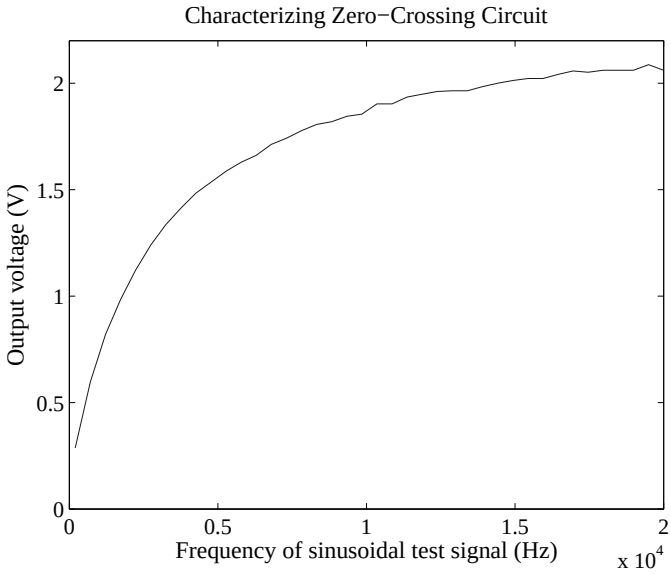


Figure 35: Measured transfer function of the zero-crossing circuit on RASP 2.8a. As the input sinusoid goes from 20 Hz to 20 kHz, the output goes from about 0.2 to 2.2 V. The response is a concave function of the number of zero-crossings, but the nonlinearity does not adversely affect classification/detection results.

convenient to implement this operation because the weights can be programmed and stored on floating-gate transistors and the final addition can be done by current summation; the circuit can be automatically generated and compiled on RASP 2.8a [93].

In our system, the features (zero-crossing rate and envelope estimation) are non-negative, but the perceptron weights (w_i) and bias (b) can be either positive or negative. Therefore, we need a two-quadrant multiplier in which the input is non-negative and the output can be either polarity. The structure to perform two-quadrant multiplication is shown in Figure 36.

The features from the feature extraction stage are in voltage mode, but the multiplier is a current-mode circuit. We use an OTA to convert the input into a current-mode signal. Again, a floating-gate-input OTA is used to minimize input offset. The reference voltage V_{ref} is selected such that I_{OTA} is non-negative for the entire range of the feature voltage. Figure 37 depicts the voltage-to-current conversion. The OTAs inherently have different transconductances because of mismatch. As a result, different transfer characteristics are observed for different channels. Because the subthreshold OTAs have a hyperbolic tangent transfer function, the conversion is nonlinear, especially for high voltage. Nevertheless, the

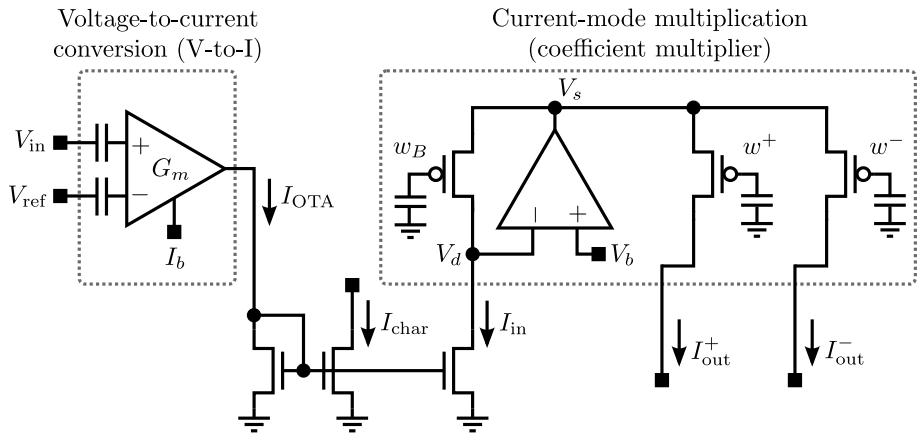


Figure 36: Building block of the vector-matrix multiplier (VMM). An operational transconductance amplifier (OTA) converts the voltage-mode sound feature into a current-mode signal. The signal is always positive. A floating-gate-input OTA is used to minimize input offset. The current-mode VMM multiplies the input signal with a weight $w = {}^{(w^+ - w^-)}/_{w_B}$. A two-quadrant multiplier is used because the input signal is positive while the weights can be both positive and negative. The differential output is $I^+_{\text{out}} - I^-_{\text{out}} = w \cdot I_{\text{in}}$. The weights are programmed onto the floating gates. The current I_{char} is used for characterization.

mismatch and nonlinearity do not affect detection performance because we can characterize the OTAs and train the perceptron with pre-distorted signals.

In the current-mode multiplier, intuitively, we use the amplifier to compress the input current such that V_s is a logarithmic function of I_{in}. Since $I^{\pm}_{\text{out}}$ is an exponential function of V_s, the logarithmic and exponential functions cancel each other. Ignoring the Early effect, the subthreshold current for a floating-gate transistor is

$$I_d = I_0 \mathrm{e}^{\frac{-\kappa V_{fg} + V_s}{U_T}}, \tag{61}$$

where I_0 is the pre-exponent term, κ is the gate-channel capacitive coupling ratio, and V_{fg} is the voltage at the floating gate. The ratio of the output and input currents is

$$\frac{I^+_{\text{out}}}{I_{\text{in}}} = \mathrm{e}^{\frac{\kappa\left(V_{fg,w_B} - V_{fg,w^+}\right)}{U_T}} = \frac{w^+}{w_B}, \tag{62}$$

where w^+ and w_B are the charges stored on the floating gates, and V_{fg,w_B} and V_{fg,w^+} are the voltages at the floating gates. Similarly, ${}^{I^-_{\text{out}}}/_{I_{\text{in}}} = {}^{w^-}/_{w_B}$. The weights are defined to be

$$w \equiv \frac{w^+ - w^-}{w_B}, \tag{63}$$

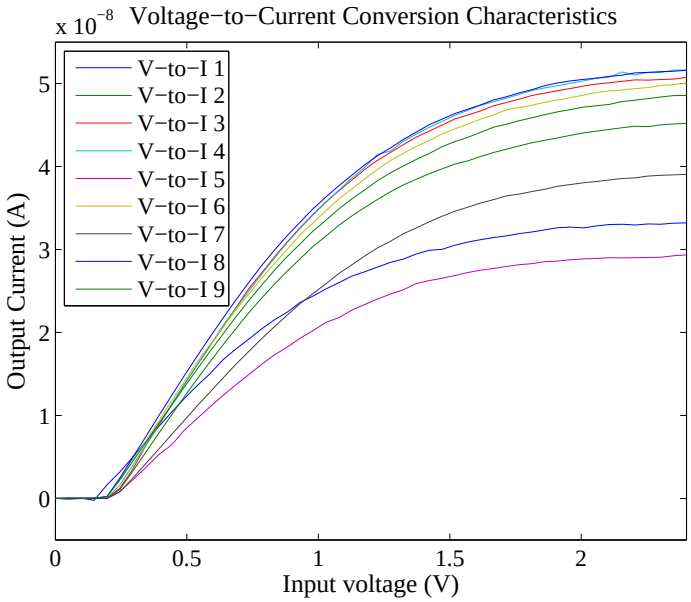


Figure 37: Transfer function of the voltage-to-current conversion on RASP 2.8a. The reference voltage, V_{ref}, is set to 2.2 V to generate a positive output current across the input voltage range. We need a positive current because the VMM structure takes a unidirectional input current. It can be observed that the OTAs inherently have different transconductances and show different transfer characteristics. In addition, the curves exhibit compression at high voltage. The mismatch and nonlinearity are not an issue because we can characterize the OTAs and train the neural network with pre-distorted signals.

and the differential output is

$$I_{\text{out}}^{+} - I_{\text{out}}^{-} = w \cdot I_{\text{in}}. \tag{64}$$

This structure produces an output current that is a scalar multiple of the input current. Since the output is a current-mode signal, we can simply tie the output nodes together to perform summation. This follows from Kirchoff's current law (KCL).

6.3.3 VMM-Based Perceptron

Combining instances of the structure in Figure 36, we can realize the expression $b + \sum_{i=1}^{8} w_i \cdot G_i(x_i)$, where G_i is the transconductance shown in Figure 37. The multiplication is achieved by the aforementioned current-mode multiplier, and the current summation follows KCL. The implementation of the perceptron is shown in Figure 38. The differential output current contains the detection result; if $i_{\text{out}}^{+} > i_{\text{out}}^{-}$ (or $I_{\text{diff}} > 0$), then the sound-of-interest is detected, and vice versa. This comparison can be easily done in a current-mode

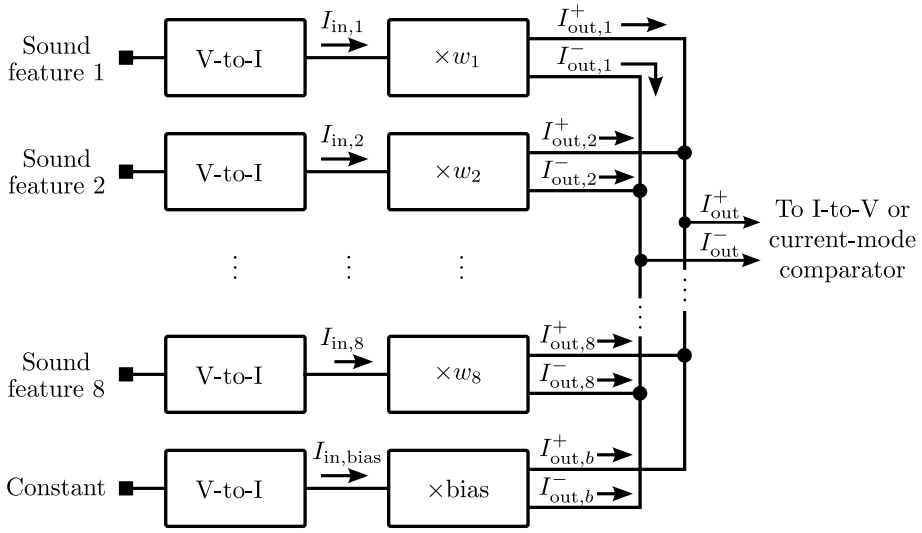


Figure 38: A VMM-based perceptron (single-layer neural network). The structure implements the expression $b + \sum_{i=1}^{8} w_i \cdot G_i(x_i)$, where G_i is the transconductance of the OTAs. The multiplication is achieved by the current-mode VMM described in Figure 36, and the current summation follows Kirchoff current law (KCL). The differential output current contains the detection result.

comparator or winner-take-all circuit (discussed in Chapter 8). The output can also be converted to voltage and digitized for subsequent processing.

6.4 Experimental Results

6.4.1 Implementation

The analog front-end can be used with many sound-detection systems. In this section, we present the implementation and experimental result for one application—glass-break sound detection. We highlight this application for two reasons: First, similar sound features (zero-crossing rate and energy) were used for glass-break sound detection, but only simulation results were presented earlier [37]. Second, power-efficient glass-break sound detection system is sought after for real-world applications, such as triggering an alarm for building and vehicle break-ins.

The availability of FPAA automation tools has helped streamline the implementation on RASP 2.8a. The circuit components were first designed in SPICE and Simulink [94]; we then used place-and-route software (GRASPER) to generate a netlist [7]; the netlist was then programmed onto RASP 2.8a with existing software and hardware infrastructures [55].

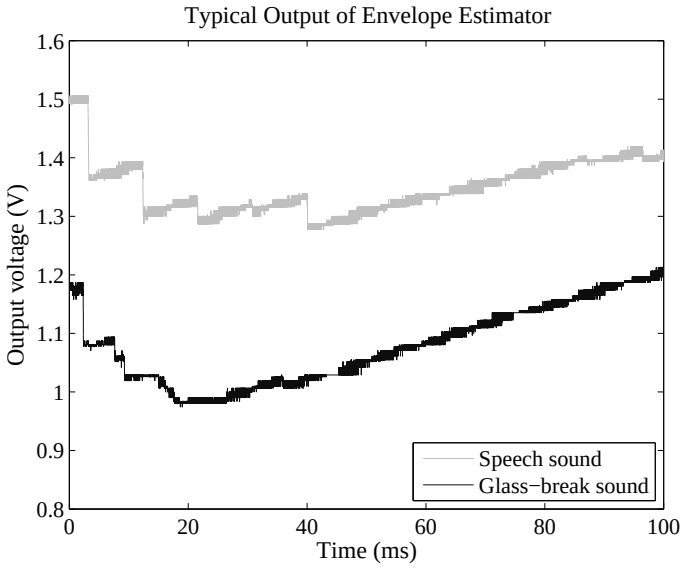


Figure 39: Typical output of the envelope detector on RASP 2.8a. The output of the envelope detection circuit for typical sub-band signals is shown. In this case, the sub-band glass-break signal has more energy than that of the sub-band speech signal and, therefore, has a lower minimum envelope. Note that the glass-break sound is not necessarily louder than the speech.

6.4.2 Training Procedure

The system was trained and tested with recorded sound samples. The database consists of 180 samples of glass-break sound and 120 samples of non-glass-break sound. The latter comprises noise, speech, and music; they are representative of typical sounds in residential and office environments. Two-thirds of the samples were randomly chosen for training; the other one-third used for testing. The training samples were processed by the feature extraction circuit, and the output was recorded and used for off-line training. The perceptron coefficients were obtained from MATLAB's Neural Network Toolbox, using Levenberg-Marquardt backpropagation [29].

6.4.3 Hardware Results

Example outputs of the feature extraction stage are demonstrated in Figure 39 and Figure 40. For the envelope detection circuit, the envelope of the sub-band glass-break sound is typically larger than that of the sub-band speech sound; therefore, the sub-band signal exhibits a lower minimum. For the zero-crossing rate circuit, the zero-crossing rate of the

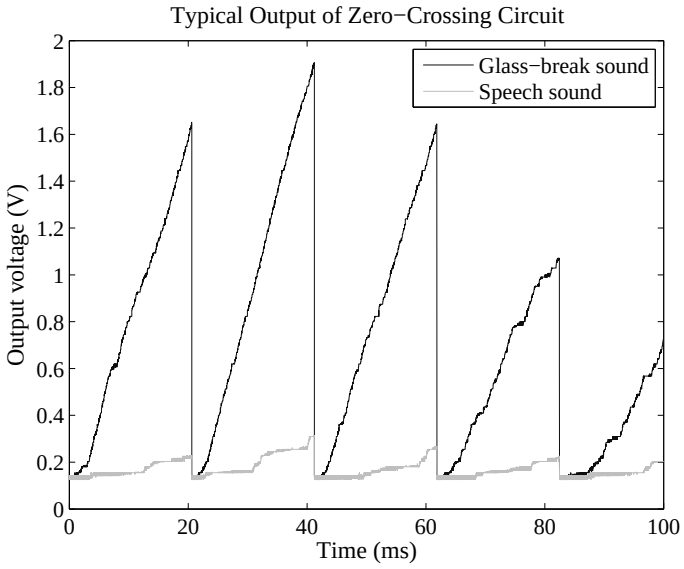


Figure 40: Typical output of the zero-crossing rate circuit on RASP 2.8a. The output of the zero-crossing circuit for typical sub-band signals is shown. In this case, the zero-crossing rate of the sub-band glass-break sound is higher than that of the sub-band speech sound. The output is sampled and reset every 20 millisecond.

sub-band glass-break sound is typically higher than that of the sub-band speech sound. The output is sampled and reset every 20 millisecond. It can be observed that at the end of every time frame, the glass-break sound has a much higher zero-crossing rate than the speech signal does. The performance of the classification stage is shown in Figure 41. The weights of the perceptron were programmed in the VMM-based structure. The input currents have different ranges because of the different transconductances (as shown in Figure 37).

The entire system was implemented on RASP 2.8a, and the experimental setup to obtain the final test results is shown in Figure 42. The place-and-route program (GRASPER) exhibited limitations in generating netlists for large systems. As a result, the feature-extraction and classification stages were implemented separately on RASP 2.8a. Using the Routing Analysis Tool (RAT), we illustrate the placement and routing topologies of the two stages in Figure 43. We measured the detection result using the aforementioned real-world sound samples. We first extracted the envelope and zero-crossing rate features from a training set. Using the training feature set, we then obtained the perceptron weights. Finally, we tested the classification stage with a test feature set and measured the output

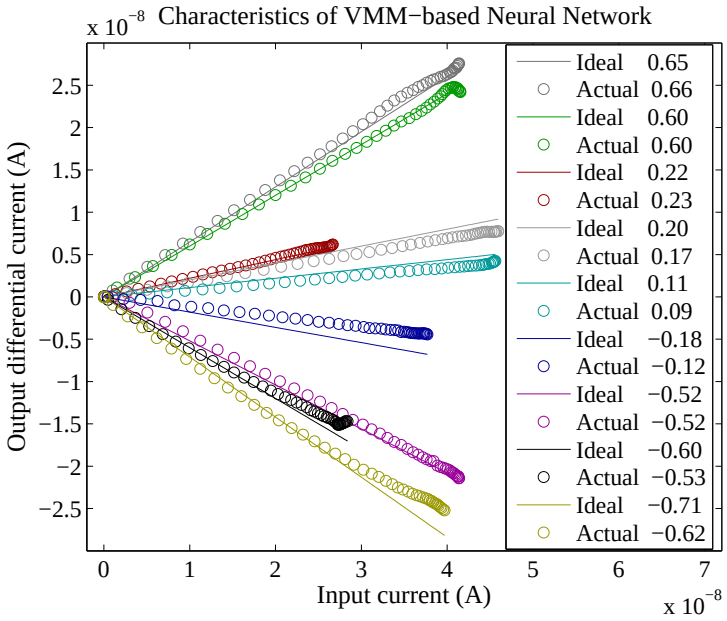


Figure 41: Coefficients of the perceptron for glass-break detection on RASP 2.8a. Both the measured and ideal coefficients are shown. The input currents have different ranges because of the different transconductance. The network was trained off-line, and the coefficients were obtained from MATLAB's Neural Network Toolbox.

currents with a picoameter. The output differential currents ($I^+_{\text{out}} - I^-_{\text{out}}$) were computed in MATLAB. The results are shown in Figure 44. It can be observed that most of the glass-break sound samples result in $I_{\text{diff}} > 0$, and vice versa. The false negative rate is 6.3%, the false positive rate 2.3%.

6.5 Discussion

We have presented the design of an analog front-end for sound detection systems. We first discussed the system-level design and algorithms. In the CADSP approach to design sensing systems, we perform some of the computation using analog signal processing before the ADC. The front-end triggers a more powerful digital system upon the detection of certain sounds. We chose sub-band envelope detection and sub-band zero-crossing rates for feature extraction. A multiple-input perceptron was used for the classification stage. We next described how we designed the circuits for different components, with a focus on employing the FPAA as an implementation platform. Finally, we presented the experimental result for a security application of glass-break sound detection.

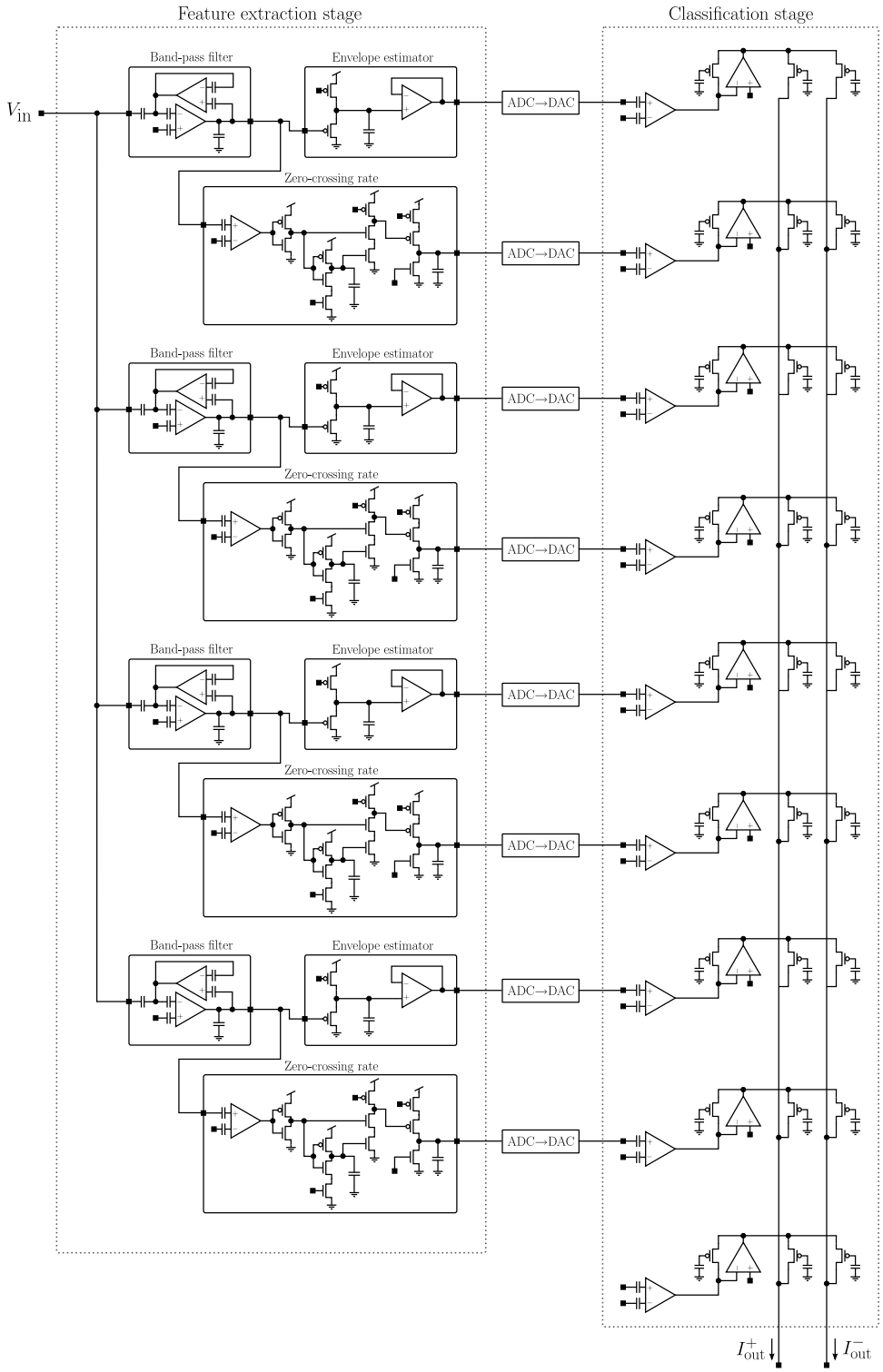
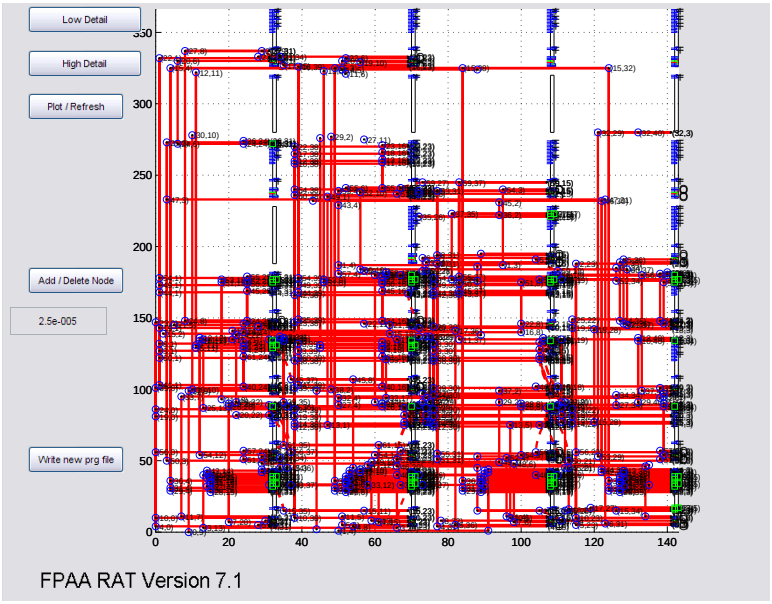
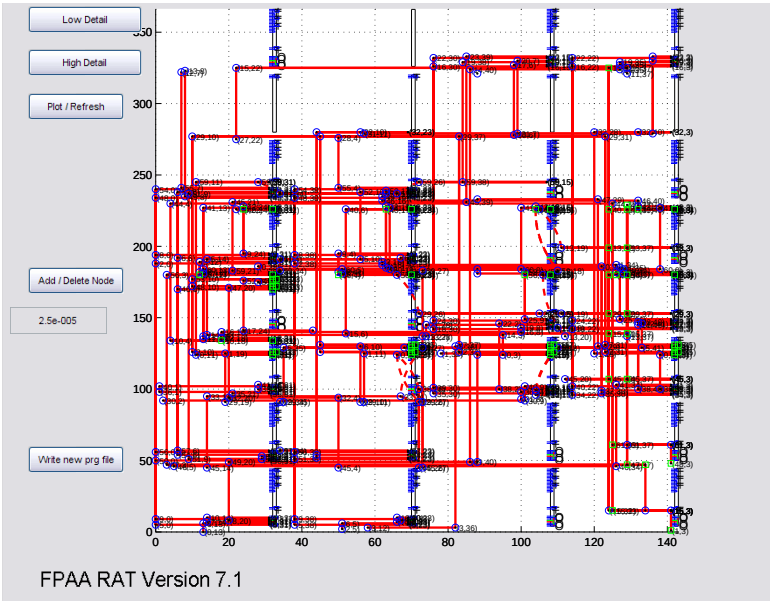


Figure 42: Experimental setup to obtain the testing results in Figure 44. Because of limitations of the place-and-route program (GRASPER), the feature-extraction and classification stages were implemented separately on RASP 2.8a. The input node V_{in} took in analog sound samples, and the output of the feature-extraction stage were recorded using on-board ADCs. The samples were fed into the classification stage using on-board DACs, and the differential output currents ($I^{\pm}_{\text{out}}$) were measured.



(a) Feature-extraction stage



(b) Classification stage

Figure 43: Placement and routing of the analog front-end on RASP 2.8a. The routing topology of the feature-extraction and classification stages as implemented on the RASP 2.8a can be visualized using the Routing Analysis Tool (RAT) [55].

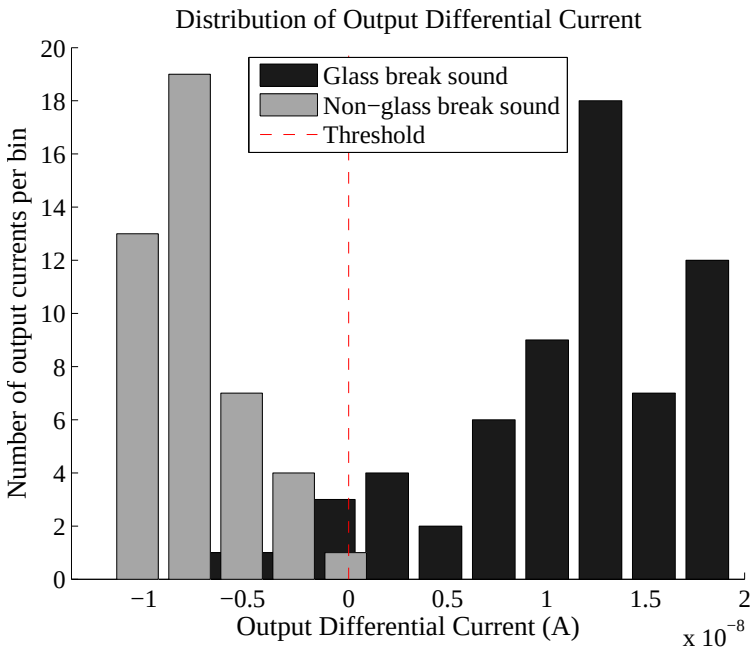


Figure 44: Glass-break detection results from RASP 2.8a. Actual sound features are fed to the detection circuit, and the differential output current is measured. The majority of the glass-break sound samples result in $I_{\text{diff}} > 0$ and vice versa. The false negative rate is 6.3%, the false positive rate 2.3%.

CHAPTER VII

DESIGNING ANALOG AUDIO CLASSIFIERS WITH ADABOOST-BASED FEATURE SELECTION

In Chapter 6, an analog front-end of a sound detector was presented. The design was based on an analog implementation of a multiple-input perceptron. The objective of this chapter is to present a method to simplify the design of the classifier front-end such that the system is more suitable for analog implementation. In particular, we focus on an analog feature selection scheme that selects a subset of the sound features most *relevant* in a given application. In this design paradigm, feature selection is enabled by AdaBoost, a well-known algorithm from the statistics and machine learning community. The use of AdaBoost is twofold: It serves as a feature selector for analog audio features, and it combines several weak classifiers into a strong one. We describe the general architecture and the algorithm to select features. The feature extractor is suitable for embedded sensing systems; it can be used as a front-end with a conventional DSP or as part of an integrated low-power analog classifier.

A system-level design example with simulation results from a TSMC-compatible 0.35-µm technology as presented in this chapter. The analog realization of this classifier architecture is the subject of Chapter 8.

7.1 Classifier Architecture

As described in Chapter 4, the audio classifier has two components: feature extraction to characterize and summarize the incoming audio signal, and classification to assign the signal into different categories based on the feature vector. The feature extraction stage is based on sub-band processing, and the classification stage combines several "base" classifiers into a "strong" classifier. The architecture of the overall system is depicted in Figure 45.

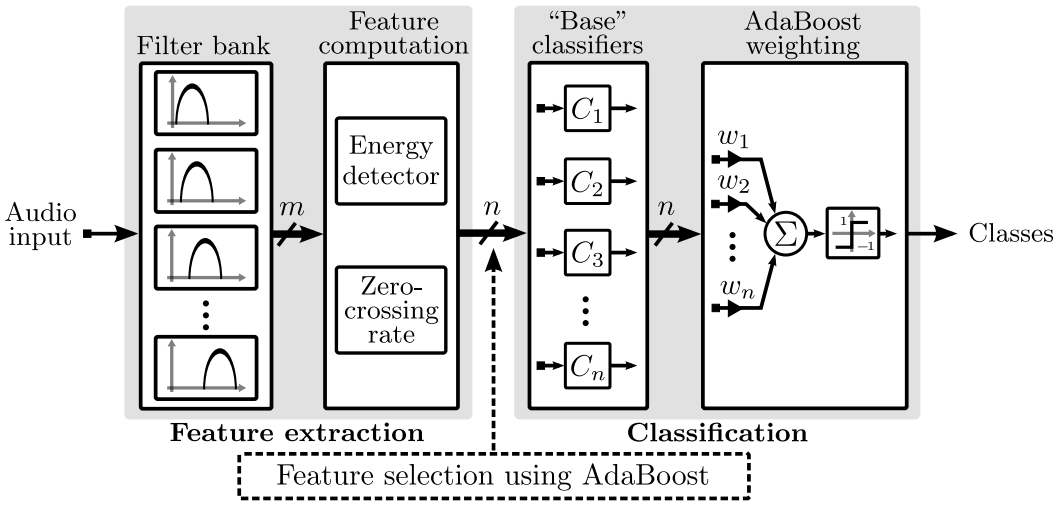


Figure 45: Architecture of the proposed audio classifier design. The classifier has two stages: analog feature extraction and classification. The feature extraction stage is based on sub-band processing: A filter bank first divides the incoming audio signal into different frequency bands; the energy and zero-crossing rate of the sub-band signals are then computed. Given the application-dependent training data, a subset of these extracted features are selected by the AdaBoost algorithm. The classification stage employs simple classifiers, such as nearest neighbor and single-input perceptron, to classify the selected features. A weighted sum of these "base" classifiers, where the weights are obtained from AdaBoost training, determines the final decision.

We have chosen *sub-band energy* and *sub-band zero-crossing rate* as audio features because simulations in MATLAB show that they are sufficient for sound detection and classification intended for embedded sensing systems, such as distinguishing speech/non-speech and detecting conspicuous sounds. The hardware implementation of these features is detailed in Chapter 6.

In the feature extraction stage, a filter bank divides the incoming audio signal into m frequency bands. The energy and zero-crossing rate of the sub-band signals are then computed, giving a feature vector of length $2m$. We then use the AdaBoost algorithm to select a subset n of these feature, where $n \leq 2m$. This feature selection process depends on the training data and is discussed in Section 7.2.

The classification stage first employs simple "base" classifiers to classify the selected

features. Three "base" classifiers are used in the experiments: nearest neighbor, single-input perceptron, and two-input perceptron. The nearest neighbor classifier is defined by

$$h_{\text{nearest}}(x_i) = \underset{k\in\{0,1\}}{\arg\min} \left(|x_i - \mu_k| \right), \tag{65}$$

where x_i is one of the extracted features, and μ_0 and μ_1 are the means of the two classes of features. In other words, h_{nn} assigns a sample to a class if the feature is closer to the mean of the feature of that class. The single-input perceptron is defined by the linear discriminant function

$$h_{1-\text{perp}}(x_i) = \begin{cases} 1 & \text{if } w \cdot x_i \geq b, \\ 0 & \text{otherwise,} \end{cases} \tag{66}$$

where x_i is one of the features, w is the weight, and b is the bias. Here, h is a one-dimensional classifier, i.e., w and x are scalars. The two-input perceptron combines features from the same sub-bands; it can be expressed as

$$h_{2-\text{perp}}(x_i, y_i) = \begin{cases} 1 & \text{if } w_1 \cdot x_i + w_2 \cdot y_i \geq b, \\ 0 & \text{otherwise,} \end{cases} \tag{67}$$

where x_i and y_i are, respectively, the energy and zero-crossing features from the i-th sub-band, and other parameters are similarly defined.

The first "base" classifier (h_{nearest}) is chosen because it was reported to be used in a *digital* implementation of audio feature selection [85]; it is included as a reference. We picked the other two "base" classifiers ($h_{1-\text{perp}}$ and $h_{2-\text{perp}}$) because they can be easily realized in both off-the-shelf systems or customized circuits, allowing low-power implementation. Specifically, the design of $h_{1-\text{perp}}$ in low-power analog circuits is discussed in the next chapter. The outputs of these "base" classifiers are weighted and summed. The sum is then compared to a threshold. The weights of this linear classifier as computed by the AdaBoost algorithm, which is discussed next.

7.2 AdaBoost and Feature Selection

The sound features included in the classifier are useful for a wide range of classification problem. However, in a specific application, some features carry more information than

Algorithm 1: The AdaBoost Algorithm [87]

Input :

1. n training examples
$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N),$$
where y_i=0 or 1 is the true class of x_i
2. T, the number of iteration
3. $h_{i,j}$, hypothesis for example x_i based on feature j

Initialize: set weights $w_{1,i} = 1/N$

for $t = 1 : T$ **do**

Normalize the weights
$$w_{i+1,i} \leftarrow \frac{w_{t+1,i}}{\sum_{k=1}^{N} w_{t+1}, k}.$$

Calculate error
$$\epsilon_j = \sum_{i=1}^{N} w_{t,i} D$$
for all the features, where $D = 0$ if $h_{i,j}$ = true class, otherwise $D = 1$.

Choose the features that corresponds to the minimum error ϵ_t. h_t is the hypothesis that corresponds to ϵ_t

Update weights
$$w_{t+1,i} \leftarrow w_{t,i} \beta_t^{e_t},$$
where $e_i = 1$ or 0 for x_i classified correctly or incorrectly, respectively, and
$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}.$$

end

others. We use a machine learning technique to select analog features based on the target class.

In Section 4.3, we discussed the resampling and model combination methods. Boosting is one such "algorithm-independent" method to improve the accuracy of any learning or classification algorithm. It combines multiple "weak" or "base" classifiers to produce a joint classifier whose performance is better than that of the individual "base" classifiers. The algorithm trains multiple classifiers in sequence, in which the error function used to train a particular classifier depends on the performance of the previous one. A popular variant of the boosting method, the AdaBoost algorithm, is used to obtain an accurate analog

audio classifier using the simple "base" classifiers. The training algorithm is described in Algorithm 1.

Originally developed for improving classifier performance, AdaBoost was modified to work as an image feature selector [101]. Ravindran and Anderson later extended the method for dimensionality reduction in audio classification in the digital domain [86]. We generalize this concept to an analog feature extraction front-end.

Two sets of training data are used to design an audio classifier. The first set of training data is used to train the "base" classifiers (h_t). These "weak" learners are then trained using the AdaBoost algorithm with the second set of data. A set of weights α_t are obtained from the algorithm. Both training procedures are currently executed off-line. The final classification is given by

$$H(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} h_t \alpha_t \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \\ 0 & \text{otherwise,} \end{cases} \tag{68}$$

where the confidence measure is $\alpha = \log {}^{1}/_{\beta_t}$ [87].

7.3 Experimental Results

7.3.1 All-Matlab simulation: Feature Extraction and Classification

The sound classifier front-end—including feature extraction and classification—was first simulated in MATLAB. The first experiment was designed to evaluate the performance of analog sound classifiers designed by the AdaBoost-based procedure outlined in Section 7.1. The objective of this experiment is to compare classifiers designed using AdaBoost against the multiple-input perceptron, which has been demonstrated in analog hardware in the previous chapter. Four classification algorithms were used in the experiments:

- a multiple-input perceptron (or single-layer artificial neural network, described in Sections 4.2.4 and 6.3.2) that takes eight features as its input—$h_{\text{neural}}(x_1, x_2, \ldots, x_8)$,
- three nearest neighbor classifiers (described in Section 4.3.1), each taking one feature, combined by AdaBoost-computed weights—$h_{\text{nearest}}(x1, x_2, x_3)$,
- three single-input perceptrons (described in Section 4.2.3), each taking one feature, combined by AdaBoost-computed weights—$h_{\text{1-perp}}(x_1, x_2, x_3)$, and

- three two-input perceptrons, each taking two features, combined by AdaBoost-computed weights—$h_{\text{2-perp}}(x_1, y_1, x_2, y_2, x_3, y_3)$.

The four classification schemes, which classify audio clips from a sound database, are depicted in Figure 46. In the experiment, the four classifiers shown in Figure 46 are tested with a sound database of five categories of sound (glass break, gunshot, speech, music, and noise), and each category is divided into three groups (A, B, and C). The classifiers are trained with two groups, and tested on the third. The average of all six permutations is reported. This arrangement is followed because the AdaBoost classifiers require two sets of training—one for the base classifiers and the other for the AdaBoost algorithm. In each test, the classifiers discriminate between two categories of sound. The MATLAB Neural Network toolbox is used to implement the neural network and the perceptron-based "base" classifiers. The training algorithm relies on random initial conditions, which lead to slightly different testing results. To compensate for the small inconsistency, ten experiments are run and the average is reported. The experimental results are summarized in Table 4. All of the AdaBoost-based classifiers have higher error rates (including both false positive and false negative errors) than the single-layer neural network does. However, while the neural network uses all eight audio features in the experiment, the AdaBoost-based designs use a subset of the features. The experimental results show that by combining simple "base" classifiers into a strong one, the AdaBoost-based designs achieve reasonable classification performance (below 10% error rate) while using fewer features. In an analog classification system, integrating more features requires additional circuitry, which consumes more power and chip space. Conversely, the AdaBoost-based classifiers consume less resources at the cost of slightly higher error rates.

Comparing the performance of the three AdaBoost-based architectures, we observe that the complexity of the underlying "base" classifiers have an impact on the performance of the combined classifier—the more powerful the "base" classifiers, the lower the final error rate. To better understand the AdaBoost algorithm, we ran a second experiment to measures its capability to enhance the performance of the "weak" classifiers. In this experiment, the performance of the combined classifiers are evaluated as more "base" classifiers (and

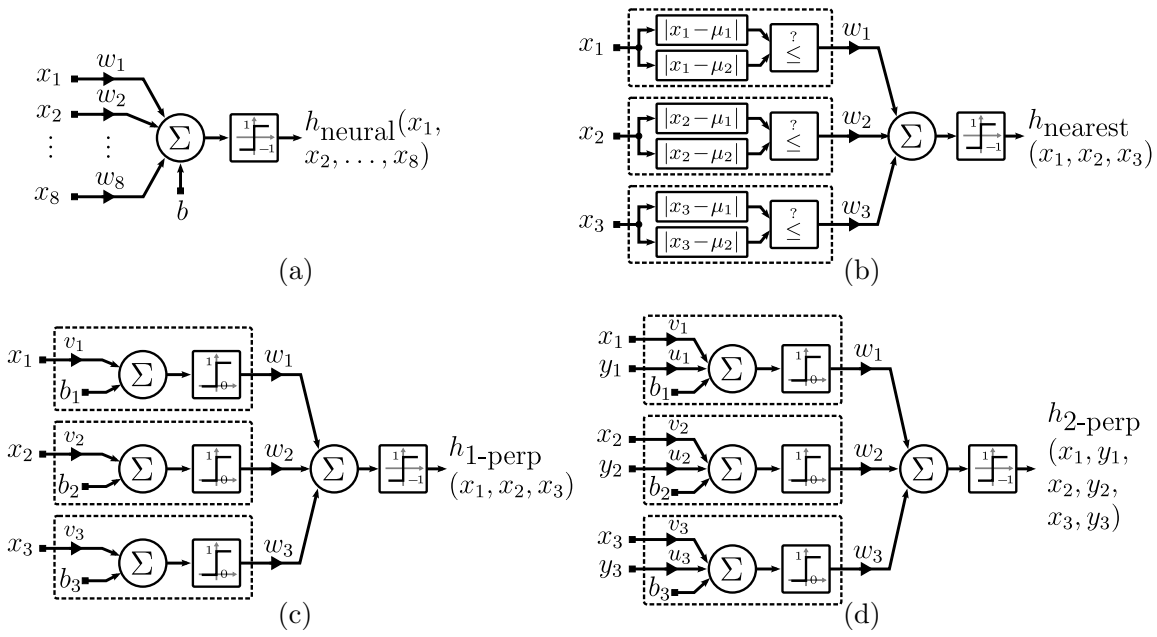


Figure 46: The four classification schemes used in the experiment. (a) Classifier $h_{\text{neural}}(x_1, x_2, \ldots, x_8)$ is a single-layer artificial neural network that takes all eight extracted features. (b) Classifier $h_{\text{nearest}}(x1, x_2, x_3)$ combines three nearest neighbor classifiers (Equation 65), each taking one features. The features are selected by the AdaBoost algorithm, which also assigns the weights. (c) Classifier $h_{\text{1-perp}}(x_1, x_2, x_3)$ combines three single-input perceptrons (Equation 66), each taking one feature selected by the AdaBoost algorithm. The weights are also assigned by AdaBoost. (d) Classifier $h_{\text{2-perp}}(x_1, y_1, x_2, y_2, x_3, y_3)$ combines three two-input perceptrons (Equation 67), each taking two AdaBoost-selected features from the same sub-band. Again, the weights are also computed by AdaBoost.

hence more features) are included. The results are plotted in Figure 47; they suggest that although incorporating more "base" classifiers generally tends to reduce the error rates, it leads to diminishing returns after three or four are included.

7.3.2 FPAA Feature Extraction and Matlab Classification Simulation

The analog feature extraction circuits described in Section 6.3 were implemented on RASP 2.8a, a programmable analog platform. RASP 2.8a, which was fabricated in a 0.35-µm CMOS process, was discussed in Section 6.2. An experiment is designed to evaluate the feature extraction circuits in the context of the AdaBoost-based classifier. We trained and tested the feature extraction circuits with the same sound database used in the previous experiments. The sound files were processed in the feature extraction circuit on RASP 2.8. The extracted features (hardware measurements expressed as voltages between 0 and 2.4

Table 4: Experimental result from MATLAB simulations. There are five types of sounds in the database: glass break, gunshot, speech, music, and noise. The classification task is to distinguish between pairs of sounds: gunshot versus speech, noise versus music, etc.

Classifier	Features used	False-positive rate	False-negative rate
h_{neural}	8	6.8%	6.8%
h_{nearest}	3	8.0%	8.0%
$h_{\text{1-perp}}$	3	8.7%	8.2%
$h_{\text{2-perp}}$	6	7.0%	7.1%

Table 5: Experimental result from FPAA measurements and MATLAB simulations. The feature extraction stage of the classifier is designed and implemented on RASP 2.8a (fabricated in a 0.35-µm CMOS process). The sound clips from the sound database are processed on and recorded from RASP 2.8a. The simulator computes the extracted features, which are used to test the four classifiers shown in Figure 46.

Classifier	Features used	False-positive rate	False-negative rate
h_{neural}	8	13.7%	13.0%
h_{nearest}	3	15.8%	15.8%
$h_{\text{1-perp}}$	3	15.7%	17.3%
$h_{\text{2-perp}}$	6	14.7%	16.9%

volts) were recorded and tested on the aforementioned four classification schemes, and the results are presented in Table 5. It can be observed that all four classifiers have higher error rates than those of the all-MATLAB simulations. This is likely due to the reduced dynamic range of the sound features in the analog implementation, which leads to a "compressed" feature space. The degradation in classification accuracy was also observed if the feature extraction stage was simulated in SPICE [23].

In this experiment, all four classifiers have similar performance, but fewer features are used in the AdaBoost-based classifiers. The result suggests that the AdaBoost-based design procedure can be used to develop sound classification systems with simple features and "base" classifiers.

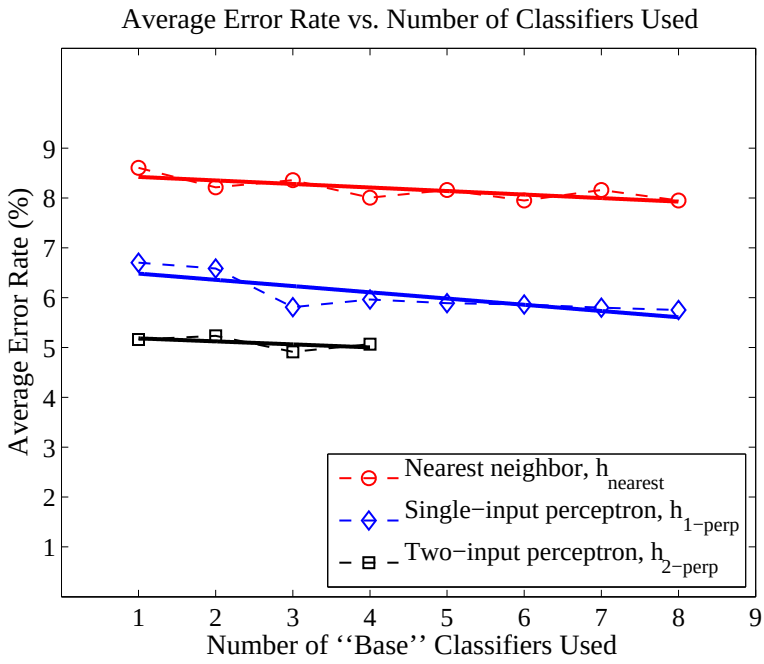


Figure 47: Average classification error rate as a function of number of "base" classifiers used. The dotted lines are simulation results, and the solid lines are linear regressions of the data. The results of this experiment show that although including more "base" classifiers generally reduces the error rates, it leads to diminishing return when more than four are used.

7.4 Discussion

An AdaBoost-based design paradigm for analog sound classifiers is presented in this chapter. The classifier architecture includes a general-purpose feature extraction front-end. The AdaBoost algorithm is used to select the most relevant features, which are processed by simple "base" classifiers. The algorithm also assigns weights for these "base" classifier, and a final decision is made based on the weighted sum of the "base" classifier outputs. Hardware measurements and MATLAB simulations demonstrate that the design procedure generates comparably accurate results as another classifier architecture of similar complexity. The current design uses a subset of the features and, as a result, it saves power and chip space.

CHAPTER VIII

ANALOG IMPLEMENTATION OF THE ADABOOST-BASED CLASSIFICATION STAGE

In Chapter 7, we described a feature selection scheme to reduce the complexity of an analog classifier front-end, and the performance of the classification stage was demonstrated via MATLAB simulation. The AdaBoost-based classification stage is essentially a two-layer design, in which the outputs of the "base" classifiers are evaluated in the AdaBoost layer. To provide an all-analog classifier front-end, we now implement the AdaBoost-based classification stage in low-power analog circuits. Two designs are presented in this chapter.

8.1 AdaBoost-Based Classifier Architecture

The AdaBoost-based classifier consists of two layers. The first layer consists of simple "base" classifiers, which are combined to form a strong classifier in the second layer. In general, there can be any number of "base" classifiers in the first layer, and the "base" classifiers can be of any type. Here, we focus on one particular AdaBoost-based classifier, in which there are three single-input perceptrons $h_{1-\text{perp}}$, whose input-output characteristic is defined as

$$h_{1-\text{perp}}(x_i) = \begin{cases} 1 & \text{if } v \cdot x_i \geq b, \\ 0 & \text{otherwise.} \end{cases} \tag{69}$$

In each perceptron, the feature value x is multiplied with a scaler (weight) v, which can be positive or negative. The result of the multiplication is compared against a threshold, which can also be positive or negative. The output of the perceptron is either 1 or 0. In the second (AdaBoost) layer, the output from the first layer is multiplied with a positive scaler. The sum of the multiplication results is compared to an AdaBoost threshold, which is positive. The overall architecture is shown in Figure 48. Both feature selection of the first layer and weight calculation of the second layer are obtained using the AdaBoost algorithm described in Section 7.2.

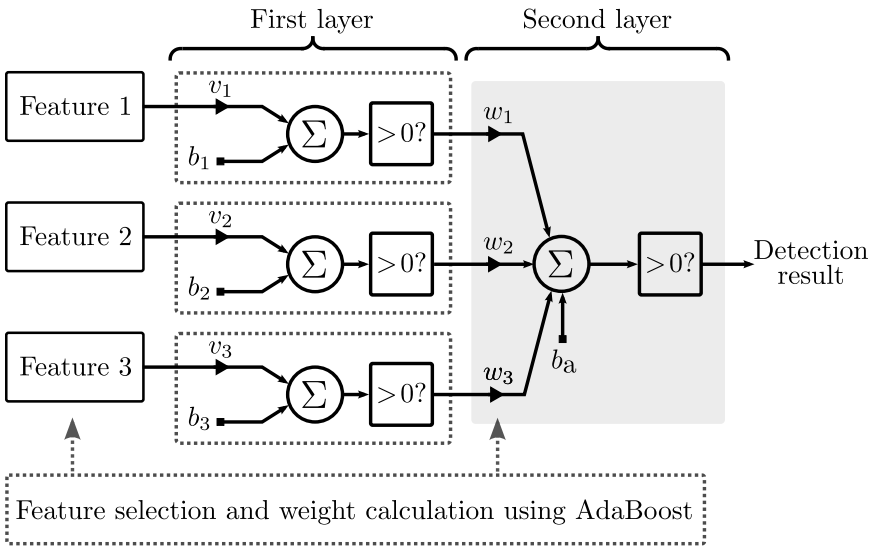


Figure 48: AdaBoost-based classifier structure. The classifier consists of two layers. There are three single-input perceptrons in the first layer. In each perceptron, the feature value is weighted and compared against a bias; the output is binary. In the second layer, the output from the first layer is weighted and compared to an AdaBoost threshold. Both feature selection and weight calculation are obtained using the AdaBoost algorithm.

8.2 VMM-based Classifier Circuit

Both the first and second layers the classifier consist of scalar multiplications and summation. Since the VMM-based multiple-input perceptron circuit presented in Section 6.3 realizes the same operations, the same VMM-based circuit can be employed in implementing the AdaBoost-based classifier. In this section, we present a design borrowed directly from the multiple-input perceptron circuit. A different design is presented in Section 8.3.

8.2.1 VMM-based Classifier Design

The implementation of AdaBoost-based classifier, as shown in Figure 49, is similar to that of the multiple-input perceptron described in Section 6.3. The perceptron in the first layer is implemented using a two-by-two vector-matrix multiplier (VMM). The differential output currents of the VMM compared in a current-mode comparator [26, 33]. Since the VMM is a current-mode structure, the input feature and bias voltages are first converted to current-mode signal using operational transconductance amplifiers (OTA). The VMM performs the multiplication and addition in Equation 69. The comparator determines whether the

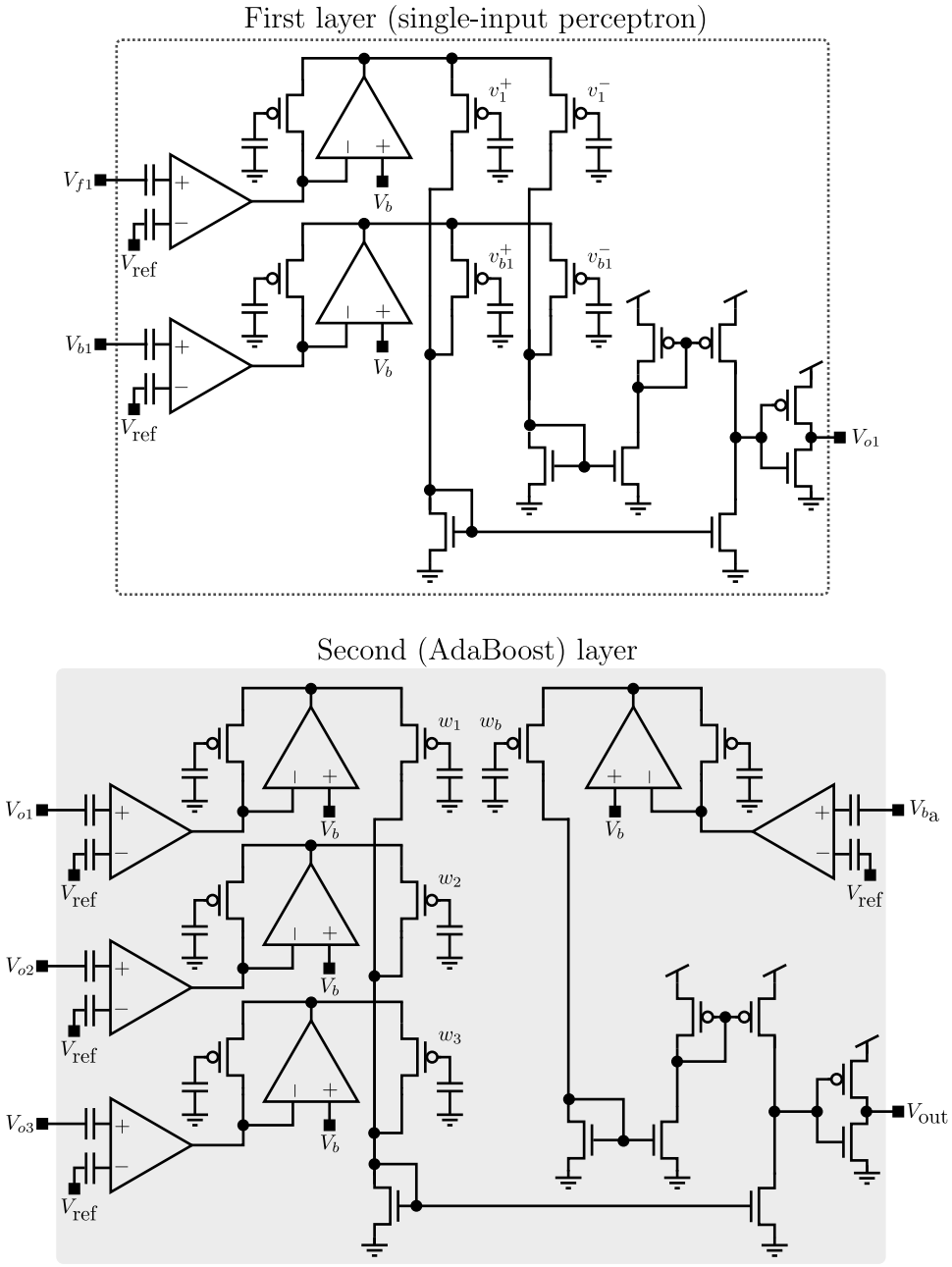


Figure 49: Implementation of the VMM-based analog classifier circuit. The perceptrons in the first layer are implemented using a vector-matrix multiplier (VMM) followed by a current-mode comparator. The input feature and bias voltages are converted to current using operational transconductance amplifiers (OTA). The VMM performs the multiplication and addition, and the comparator determines if the differential current is above or below zero. The inverter output is "high" if $v_1 \cdot V_{f1} > V_{b1}$. The second (AdaBoost) layer performs a weighted summation of the first layer outputs. The OTAs in the second layer are programmed such that their respective output current is either zero or "high," depending on the inverter output. The output voltage V_{out} is "high" if $\sum_{i=1}^{3} w_i \cdot V_{oi} > V_{b_\text{a}}$, and vice versa. The summation is performed as a result of Kirchoff's current law (KCL). Single-sided VMM is used in the second layer because all the weights are positive.

differential current is positive or negative—the inverter output is "high" if $v_1 \cdot V_{f1} > V_{b1}$, and vice versa. The input signal is always positive, but the weights can be positive or negative. Therefore, we use a single-ended-input, differential-output VMM to implement the perceptron.

The second (AdaBoost) layer is similar to the first layer; however, since the weights are all positive, we use single-ended-input, single-ended-output VMM to implement this layer. In the second layer, we perform a weighted summation of the first layer outputs. The OTAs in the second layer are programmed such that their output current is binary. The output current is considered zero (in picoamps) if the inverter output is "low," and it is non-zero (in tens of nanoamps) if the inverter output is "high." The output voltage of the AdaBoost layer is given by

$$V_{\text{out}} = \begin{cases} \text{“high”} & \text{if } \sum_{i=1}^{3} w_i \cdot V_{oi} > V_{b_{\text{a}}}, \\ \text{“low”} & \text{otherwise.} \end{cases} \tag{70}$$

The summation is performed as a result of Kirchoff's current law (KCL), and the comparison (thresholding) is realized using a current-mode comparator.

8.2.2 Experimental Results of the VMM-based Classifier

To investigate the feasibility of implementing the VMM-based classifier circuit on analog hardware, we simulated the circuit of Figure 49 in SPICE using transient analysis. We ran the simulation with a 0.35-µm CMOS process, on which the RASP 2.8a chip is fabricated [9].

An important circuit element in the classifier design is the floating-gate transistor (see Section 6.2), but there is no standard SPICE model for this element. To simulate the floating-gate transistors (both the CAB and switch elements) on the FPAA, we adopted the floating-gate transistor model in [97]. As illustrated in Figure 50, the DC voltage V_{prog}, set by a diode-connected PFET, simulates the voltage on the floating-gate node due to the stored charge. This voltage is connected to the floating-gate through an very large resistor (in petaohms), so almost no current flows through the floating-gate node. The floating-gate voltage V_{fg} is contributed by both V_{prog} and the gate voltage V_{g}. In the "run mode" of the RASP 2.8a chip, V_{g} is a global (chip-wide) parameter and typically set to zero. As a result,

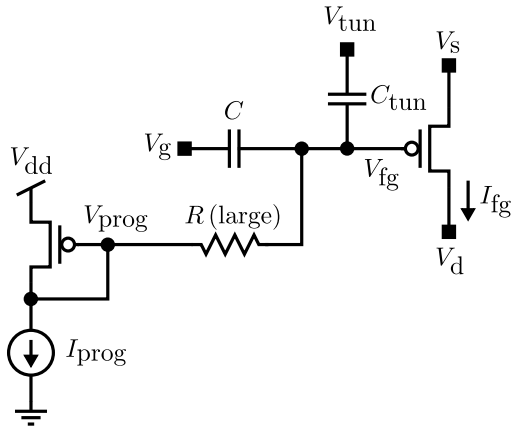


Figure 50: Floating-gate transistor model [97]. The DC voltage V_{prog}, set by a diode-connected PFET, simulates the charge stored on the floating-gate node. This voltage is connected to the floating-gate through a very large resistor (about $1 \times 10^{15}\ \Omega$), so practically no current flows through this node. The floating-gate voltage V_{fg} is a combination of V_{prog} and the gate voltage V_{g}. In the "run mode" of the FPAA, V_{g} is typically set to zero. As a result, I_{prog} determines the floating-gate current I_{fg}.

the floating-gate current I_{fg} is set by I_{prog}.

In this experiment, we tested the capability of the VMM-based classifier circuit to detect glass-break sound (cf. Section 6.4.3). A sound database consisting of 180 samples of glass-break sound and 120 samples of non-glass-break sound is used in training and testing the AdaBoost-based classifier. The non-glass-break sound comprises noise, speech, and music; they are representative of typical sounds in residential and office environments. Both the training and testing sound samples were processed by the feature extraction circuit discussed in Chapter 6. Eight sound features were used to represent input audio signal—four sub-band energy estimation and four sub-band zero-crossing rate. The feature output was recorded and used for off-line training and testing.[1] One-third of the sound samples were randomly chosen to train the "base" classifiers (the single-input perceptrons). The perceptron coefficients were obtained from MATLAB's Neural Network Toolbox, using Levenberg-Marquardt backpropagation [29]. The other one-third of the sound samples were used to test the "base" classifiers using the AdaBoost algorithm (Section 7.2). The algorithm selected the three features that, in combination with the single-input perceptron, produces the lowest classification errors; it also assigned the weights for the "base" classifiers, to be used in the

[1]The feature extraction stage was performed in hardware (RASP 2.8a), not SPICE simulation.

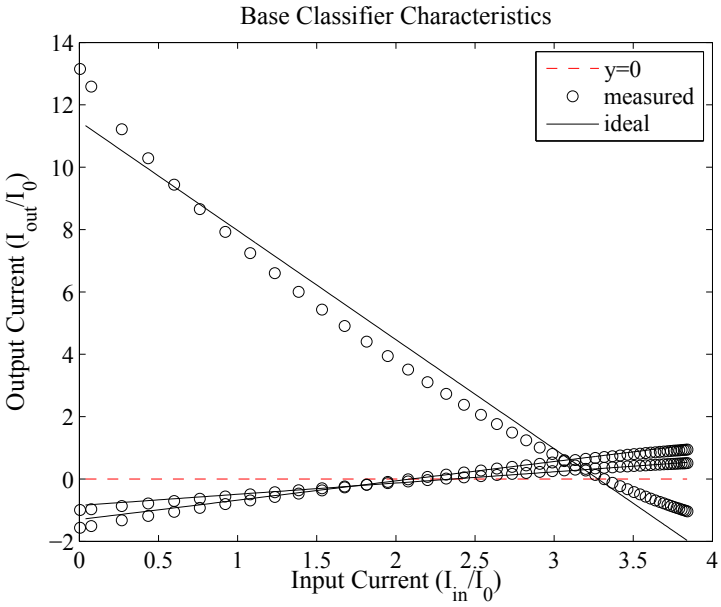


Figure 51: Characterizing the "base" classifiers in the VMM-based circuit. The transfer functions of the single-input perceptrons are shown. They can be expressed by a linear function $V_y = v \cdot V_x + b$. When followed by a comparator, the perceptron output is "high" if $V_y \geq 0$, and vice versa. The measured and ideal curves are similar, especially near the threshold $V_y = 0$.

second layer.

We first characterized the first layer of the AdaBoost-based classifer—the three "base" classifiers. The transfer functions of the single-input perceptrons are shown in Figure 51. They can be expressed by a linear function $V_y = v \cdot V_x + b$, where v and b are coefficients of the perceptron, and V_x is the input signal (in voltage). When followed by a current-mode comparator, the perceptron output is "high" if $V_y \geq 0$, and vice versa. The figure illustrates that the measured and ideal curves are similar, especially near the threshold $V_y = 0$. The curves are plotted in normalized form, and the normalization factor is $I_0 = 20$ nA. In absolute terms, the floating-gate transistors on the input-legs of the VMM were programmed to 100 nA. For large coefficients, the output current of the VMM can be as high as hundreds of nanoamps. In this current range, the floating-gate transistors are at the edge of the subthreshold region. This is likely to cause the measured current to deviate from the ideal curve at high output current. The second layer also performs a weighted summation followed by a current-mode comparator, and the measurement for the second

Table 6: Confusion matrix of the VMM-based classifier circuit SPICE simulations. The AdaBoost-based classifier was tested with a sound database consisting of glassbreak and non-glassbreak sound. The (first layer) "base" classifiers were first trained with a set of sound samples. The (second layer) AdaBoost classifiers were trained with a second set of sound samples. The entire system was then tested with a third set of samples. The false positive rate is 16%, and the false negative rate is 1.6%.

		Predicted class	
		Glassbreak	Non-glassbreak
Actual class	Glassbreak	62	6
	Non-glassbreak	1	38

layer is similar to that of the first layer.

While the two-thirds of the sound samples in the database were employed to train the first and second layers of the AdaBoost-based classifier, the other one-third of the samples were used to test the classifier in SPICE. The glass-break sound detection results are presented in Table 6. The false positive rate is 16%, and the false negative rate is 1.6%. The error rates are higher to those obtained from an AdaBoost-based classifier simulated in MATLAB.

8.3 Simplified AdaBoost-based Classifier Circuit

Although the VMM-based design can be realized on analog hardware (e.g., custom IC or FPAA), it can be simplified for a more efficient implementation. In this section, we present a different classifier circuit to realize the AdaBoost-based classifier. In particular, we exploit two properties of the classifier architecture (Figure 48). First, the single-input perceptron is essentially a thresholding function; second, the output of the first layer is digital (either "1" or "0"). This simplified design is implemented on an FPAA, and hardware results are reported.

8.3.1 Simplified AdaBoost-based Classifier Design

The implementation of the first layer of the AdaBoost-based classifier is shown in Figure 52. The single-input perceptron is essentially a linear function followed by a step function. Its

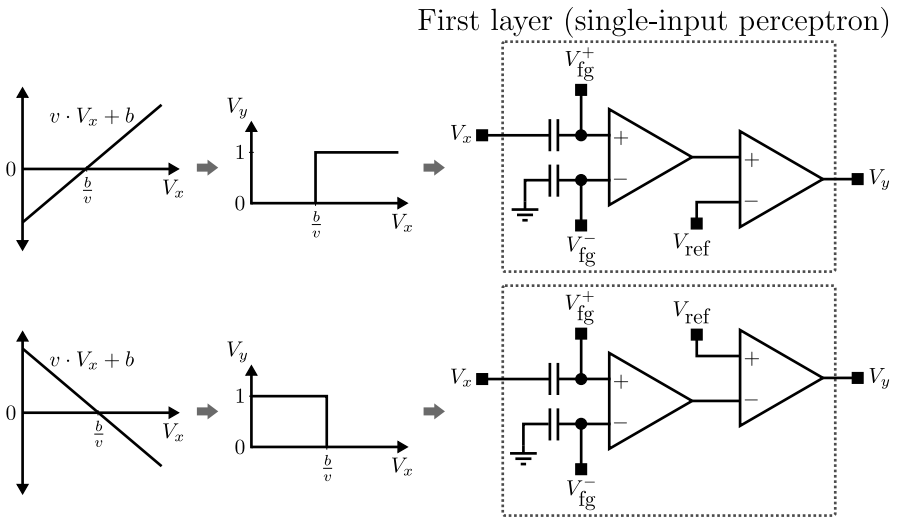


Figure 52: Implementation of the first layer of the AdaBoost-based classifier on the FPAA. The transfer function the single-input perceptron can be expressed as a shifted step function $V_y = U\left(\text{sgn}(v) \cdot x - {}^{b}/_{v}\right)$, where $U(\cdot)$ is a unit step function at the origin, and $\text{sgn}(\cdot)$ is the signum function. This function can be realized by an open-loop floating-gate-input OTA, which has the transfer function $V_y = U\left(\left({}^{C_{\text{in}}}/_{C_{\text{tot}}}\right) V_x + V_{\text{fg}}^+ - V_{\text{fg}}^-\right)$, where ${}^{C_{\text{in}}}/_{C_{\text{tot}}}$ is the input capacitive divider, and V_{fg}^+ and V_{fg}^- are due to charge programed to the positive and negative input terminals, respectively. The second open-loop OTA changes the "direction" of the step function and sharpens the output transition.

transfer function can be expressed as

$$V_y = U\left(\text{sgn}(v) \cdot x - \frac{b}{v}\right), \tag{71}$$

where $U(\cdot)$ is a unit step function at the origin, and $\text{sgn}(\cdot)$ is the signum function. When the weight (slope) v is positive, V_y is a shifted step function; when v is negative, V_y is input-reversed, shifted step function. This function can be realized by an open-loop floating-gate-input OTA, which has the transfer function

$$V_y = U\left(\frac{C_{\text{in}}}{C_{\text{tot}}} V_x + V_{\text{fg}}^+ - V_{\text{fg}}^-\right), \tag{72}$$

where ${}^{C_{\text{in}}}/_{C_{\text{out}}}$ is the attenuation factor introduced by the input capacitive divider (about ${}^{1}/_{11}$ on the RASP 2.8a), and V_{fg}^+ and V_{fg}^- are, respectively, due to the charge programed to the floating gates at the positive and negative input terminals. The floating-gate charges are adjusted such that their difference $\frac{C_{\text{tot}}}{C_{\text{in}}}\left(V_{\text{fg}}^+ - V_{\text{fg}}^-\right)$ is equal to $-{}^{b}/_{v}$. The second open-loop OTA changes the "direction" of the step function—the polarities of the input terminals

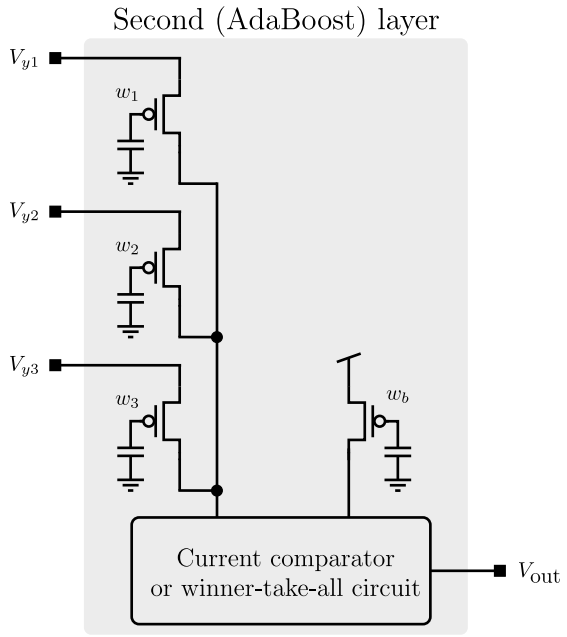


Figure 53: Implementation of the second layer of the AdaBoost-based classifier on FPAA. The outputs of the first layer, which are either one or zero, are weighted, summed and compared to the AdaBoost threshold. The binary outputs of the first layer directly connect to the source of the floating-gate transistors, which are programmed with the AdaBoost weights. KCL allows the currents to be added together. A current-mode comparator or winner-take-all circuit can be used to realize the comparison.

depend on the sign of v. The second OTA also supplies additional gain to sharpen the voltage transition at the output.

The implementation of the second layer of the classifier is shown in Figure 53. Here, the outputs of the first layer are weighted, summed and compared to the AdaBoost threshold. Since the first layer outputs V_y are either one or zero, they either turn on or off the associated weight. To implement this binary operation, we connect the (voltage mode) outputs of the first layer to the source of floating-gate transistors. The floating-gate transistors are programmed to carry currents that are proportional to the weights. When V_y is "high" (at V_{dd}), the floating-gate transistor acts as a current source; when it is "low" (at ground), the transistor is switched off. Ideally, the source of the floating-gate transistor should be floating (at high impedance) when we intend to switch it off because if the input node V_y is at ground, the drain current could flow the other way into the input node. This situation can be prevented if the drain of the floating-gate transistors are connected to a low-impedance path to ground, as is the case. The summation follows as a result of KCL. The AdaBoost

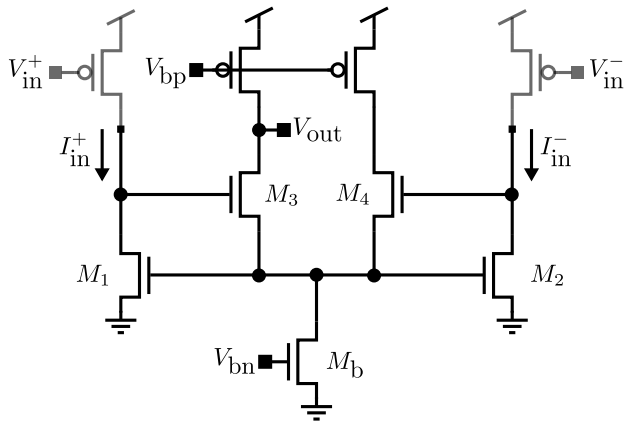


Figure 54: Winner-take-all circuit used as a current comparator. A two-cell current-mode winner-take-all (WTA) circuit is shown. The shaded PFETs are used for characterizing the circuit. In "run mode," the input currents are supplied by the second layer of the AdaBoost-based classifier. The circuit is biased to run in subthreshold ($V_{\text{bn}} \approx 0.5$ V and $V_{\text{bp}} \approx 1.8$ V in a 0.35-µm process). V_{out} goes "low" if $I_{\text{in}}^{+} > I_{\text{in}}^{-}$, and vice versa.

threshold is simply implemented as a floating-gate current source, which is programmed to carry a current proportional to the threshold.

A current comparator or winner-take-all (WTA) circuit can be used to realize the comparison between the weighted sum and the AdaBoost threshold. The winner-take-all circuit is chosen in this design because it can be generalized to include additional inputs in the second layer for multiple-class classification. A two-cell current-mode winner-take-all circuit is shown in Figure 54 [62]. The shaded PFETs are used for characterizing the circuit; in "run mode," the input currents are supplied by the second layer. When the input currents are equal ($I_{\text{in}}^{+} = I_{\text{in}}^{-}$), M_1 and M_2 would have the same current, and M_3 and M_4 would have the same current; as a result, the drain voltages of M_3 and M_4 are the same. If $I_{\text{in}}^{+} > I_{\text{in}}^{-}$, M_1 would remain in saturation, but M_2 would go into the Ohmic region. Consequently, M_4 is switched off and M_3 carries all the current supplied by M_b. In this case, V_{out} gets close to ground ("low"). Conversely, if $I_{\text{in}}^{+} < I_{\text{in}}^{-}$, M_3 is effectively switched off, and V_{out} becomes close to V_{dd} ("high").

8.3.2 Experimental Results of the Simplified Classifier

The simplified classifier circuit was implemented on a field programmable analog array. The implementation platform, RASP 2.8a, was discussed in Section 6.2. We tested the

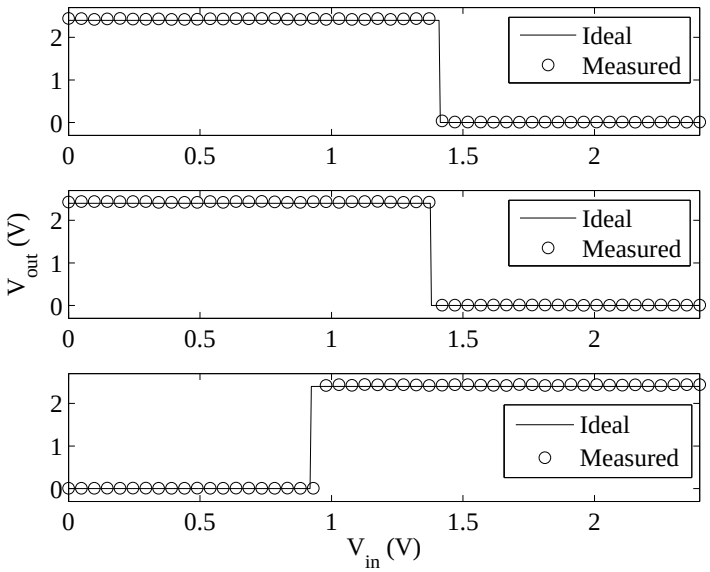


Figure 55: Experimental result for the first-layer single-input perceptrons. The ideal and measured transfer functions of three single-input perceptrons are shown. The transfer functions are essentially shifted step functions, and the transition occurs at the y-intercept of the linear function that defines the single-input perceptron. The sign of the weight determines whether the step function has a low-to-high or high-to-low transition.

AdaBoost-based classifier with the same glass-break detection application discussed in Sections 6.4 and 8.2.2.

We first characterized the single-input perceptrons; the experimental results are shown in Figure 55. The floating-gate-based OTAs in Figure 52 were programmed with a set of coefficients, which correspond to the weight and bias of the perceptrons. The transfer functions are shifted step functions, and the ideal and measured traces are similar. The floating-gate transistors in the second layer were programmed to carry currents proportional to the AdaBoost weights. The weights (less than or equal to one) were normalized to 50 nA, so that the currents were small enough to be in subthreshold but large enough to achieve accurate programming. The characterization of the winner-take-all circuit is shown in Figures 56 and 57. The circuit has a fairly high gain and small offset.

The first and second layers of the classifier were implemented on the RASP 2.8a chip, and the experimental setup to obtain the final testing results is shown in Figure 58. A database of glass-break and non-glass-break sound samples were used to train and test the classifier.

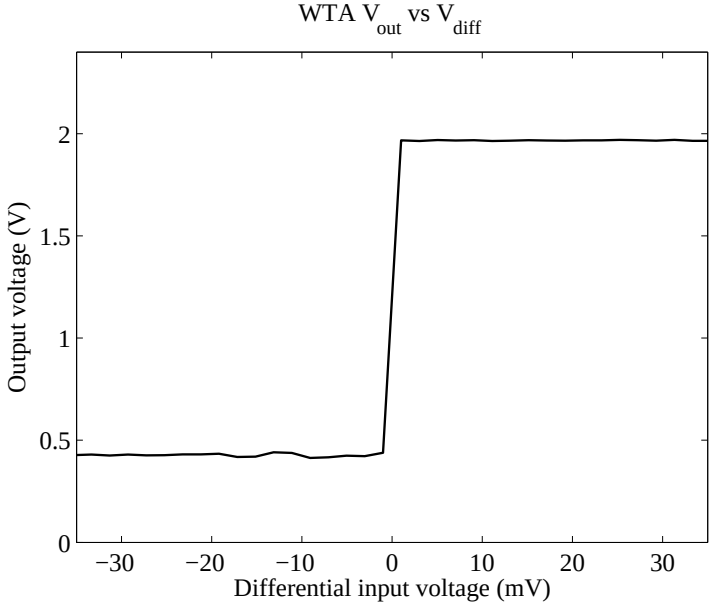


Figure 56: Winner-take-all circuit output voltage vs. differential input voltage. We characterized the winner-take-all circuit with the shaded PFETs in Figure 54. The output voltage is plotted against the differential input voltage. The circuit has a fairly high gain, and the output voltage has a sharp transition.

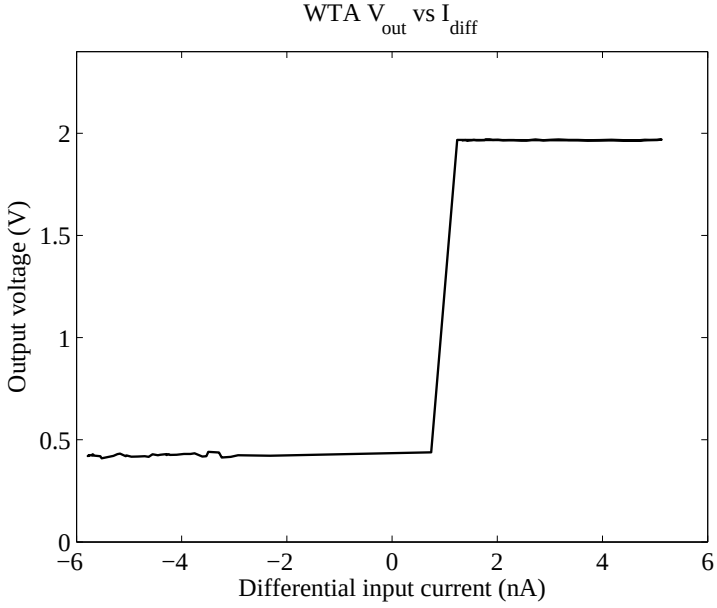


Figure 57: Winner-take-all circuit output voltage vs. differential input current. The input current is measured between $V_{\rm dd}$ and the source of the shaded PFETs (while $V_{\rm in}^{\pm}$ is being swept). There is a small offset in the transition of the output voltage; it is due to device mismatches. The offset may also be caused by driving M_3 or M_4 (in Figure 54) into Ohmic region.

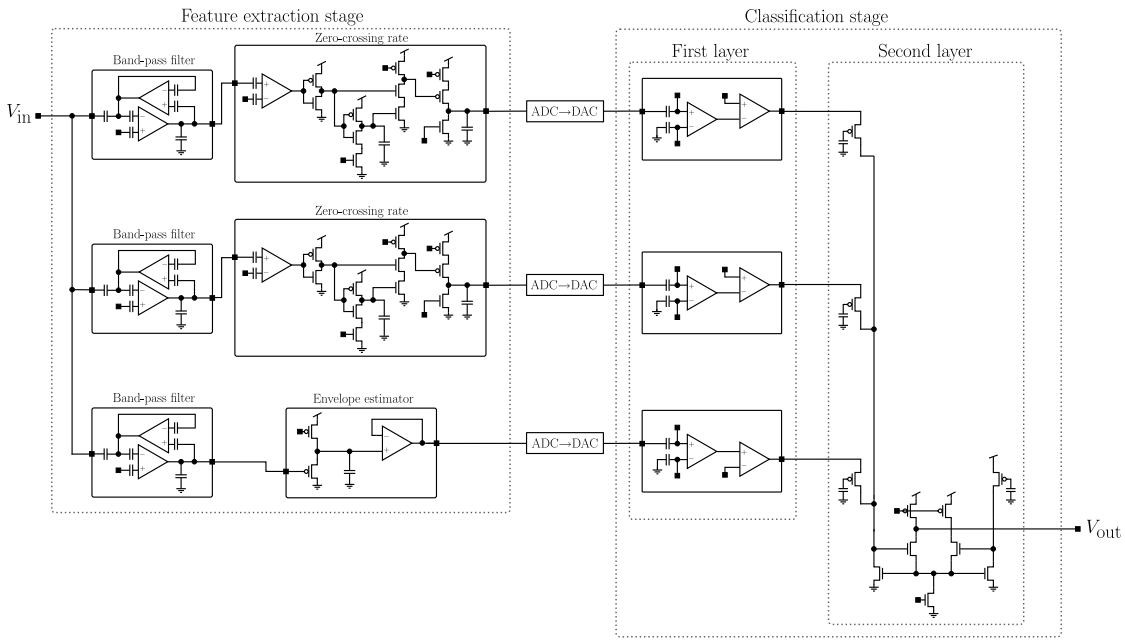


Figure 58: Experimental setup to obtain the testing results in Figure 60. The first and second layers were implemented on the RASP 2.8a chip and were tested with three sound features selected by the AdaBoost algorithm. The feature-extraction stage was discussed in Section 6.3; it was implemented separately on the same FPAA, and the sound features of the entire sound database were processed and recorded. To test the classifier circuit, we used on-board DACs to feed the sound features of each sample to the first layer. The output voltage of the second layer (V_{out}) was then measured.

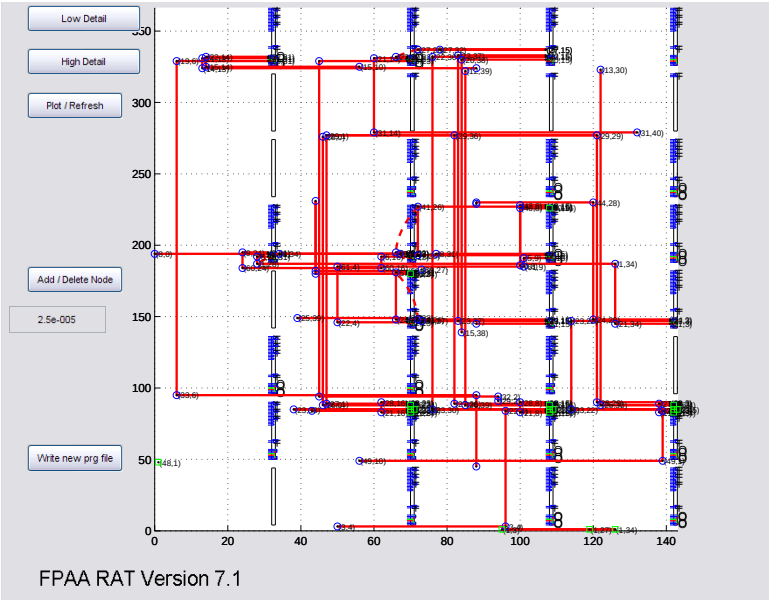


Figure 59: Placement and routing of the AdaBoost-based classifier on RASP 2.8a. The routing topology of the classification stage as implemented on the RASP 2.8a can be visualized using the Routing Analysis Tool (RAT) [55]. Only a small portion of the FPAA is utilized (cf. Figure 43(b)).

The sound samples had been processed with the feature extraction circuit discussed in Section 6.3, and the sound features were recorded using on-board ADCs. The sound samples were divided into three groups. The sound features of the first two groups were used to train the first and second layers of the classifier, respectively. We used MATLAB's Neural Network toolbox to obtain the perceptron coefficients. We then used the AdaBoost algorithm to choose three most relevant "base" classifiers and assign weights for them. We programmed the classifier circuit—including the classifiers coefficients obtained from off-line training—on the RASP 2.8a chip. Figure 59 illustrates the placement and routing topology of the classification stage as implemented on the RASP 2.8a chip. To test the classifier circuit, we used on-board DACs to feed the sound features of each sample to the first layer, and the output voltage of the second layer (V_{out}) provided the classification results.

The test results of detecting glass-break sound with the classifier circuit are shown in Figure 60. The output voltage of the winner-take-all circuit is either near V_{dd} ("high") or near ground ("low"). If the mid-rail voltage (1.2 V) is taken as the decision threshold, most of the glass-break sound is classified as "low," and non-glass-break sound "high." An inverter can be used to force the output to rail voltages. A total of about 100 samples were used in the test. The false negative rate is 6.3%, and the false positive rate is 4.5%. This classification result is similar to that of the multiple-input perceptron reported in Chapter 6. Even though a two-layer classification stage is used, the design is simpler. More importantly, we use only a fraction (3/8) of the sound features to achieve comparable classification results, hence saving power and chip space.

More test results are illustrated in Figure 61. In this test, we programmed the AdaBoost-based classifier to detect speech and music from environmental noise. This is an important task in modern smart phones, on which the users often run applications to classify music (e.g., Shazam®) and log speech (e.g., for "lifelogging"). Utilizing the sound detection front-end, we can program the smart phones such that these applications (which often run on powerful digital processors) are activated upon the detection of speech and music. The training procedure is the same as that of the glass-break detection test, and a variety of speech, music, and environmental noise sounds are included in the training and testing

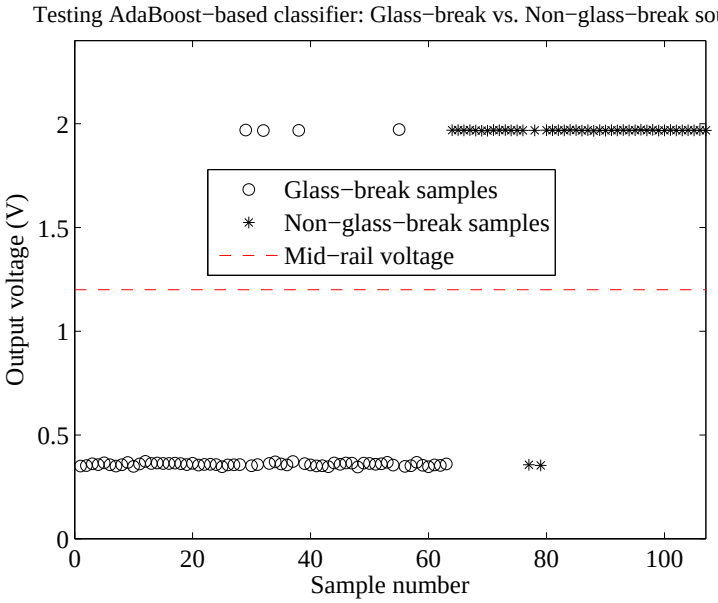


Figure 60: AdaBoost-based classifier test results. The AdaBoost-based classifier is tested with glass-break and non-glass-break sound samples. A subset of the features (as selected by the AdaBoost algorithm) is fed to the first layer. The output voltage of the winner-take-all circuit is either near V_{dd} or near ground. If we use the mid-rail voltage (1.2 V) as the decision threshold, most of the glass-break sound is classified as "low," and non-glass-break sound "high." With about 100 testing samples, the false negative rate is 6.3%, and the false positive rate is 4.5%.

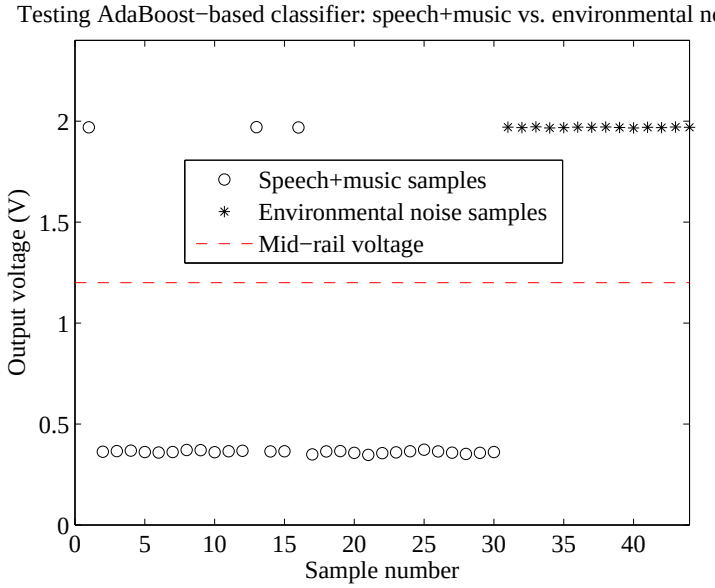


Figure 61: More AdaBoost-based classifier test results. The AdaBoost-based classifier is tested with speech and music against environmental noise. This test demonstrates that the classifier can be used as the front-end of smart phones that run music classification and speech logging applications. Upon the detection of speech or music, the more powerful applications are activated. About 45 test samples were used; the false negative rate is 10%, and there is no false positive error.

samples. About 45 test samples were used; the false negative rate is 10%, and the false positive rate is 0.

8.4 Discussion

Two designs of the AdaBoost-based analog classifier are presented. In the first design, VMMs and current-mode comparators are used to implement both layers because the algorithm calls for weighted summation and thresholding. It is an extension of the multiple-input perceptron presented in Chapter 6. In the second design, we simplify the circuit by exploiting two properties of the classifier architecture. Floating-gate OTAs are used to implement the single-input perceptrons. Since the first layer outputs are binary, we switch on and off the floating-gate-based current sources in the second layer. A winner-take-all circuit is used to perform the final comparison. The hardware test results of the simplified classifier circuit are better than the simulation test results of the VMM-based classifier circuit. The improvement is due to the reduced complexity, especially in eliminating the nonlinear voltage-to-current conversions. The second design can be generalized to implement a larger classification stage with more "base" classifiers in the first layer.

CHAPTER IX

SUMMARY AND EXTENSIONS

9.1 Summary of Contributions

In this research, we made contributions in two categories: psychoacoustic signal processing and cooperative analog-digital signal processing. In the first category, we exploit properties of the human auditory system to efficiently process audio signals; in the latter, we use low-power analog circuits to efficiently process the signal before digitizing it.

- Psychoacoustic Signal Processing

    1. A sound enhancement algorithm was designed to make piezoelectric loudspeakers sound "richer" and "fuller." Piezoelectric speakers have a small form factor but exhibit weak response in the low-frequency region. In this algorithm, we combine psychoacoustic bass extension (virtual pitch) and dynamic range compression (reducing peak-to-RMS ratio) to improve the perceived bass coming out from the speakers.
    2. The piezoelectric speaker sound enhancement algorithm was demonstrated on a real-time DSP platform. We implemented the algorithm on a Texas Instrument floating-point DSP (TMS320C6713). Using the real-time system, we conducted subjective testing to show that the algorithm improves the "richness" and overall quality of music and speech signals.
    3. An audio power reduction algorithm was developed for loudspeaker power management applications. The perceptually transparent algorithm simultaneously extends the battery life of mobile devices and prevents thermal damage in speakers. In designing this algorithm, we were inspired by audio compression algorithms, which encode audio signals in such a ways that the compression artifacts are not easily perceivable. Instead of reducing the storage space (information

content), however, we reduce the portion of the audio signal that are below the hearing threshold, suppressing inaudible contents from the audio stream before sending it to the amplifier and speaker.

- Cooperative Analog-Digital Signal Processing

    1. An analog front-end for sound detection was designed. The system embodies the CADSP design paradigm, in which computation is performed in low-power analog circuits before the signal is digitized—it is an example of an analog-to-information converter. Unlike most of the previously reported analog classifiers, the sound detector front-end is not an ad hoc design; rather, it can be used in a wide range of applications because programmable floating-gate transistors are employed to store classifier weights.

    2. The sound detector front-end was demonstrated on a reconfigurable analog platform. We designed and implemented the feature extraction and classification stages of a glass-break sound detection circuits on the RASP 2.8a chip, a field programmable analog array for general purpose signal processing applications. Although many subcomponents of the classifier circuit had been demonstrated on the FPAAs, we built a sophisticated analog system that attained the limit of the place-and-route program's capabilities.

    3. A feature selection algorithm was incorporated in the design of an analog classifier. To simplify the analog sound detection front-end, we utilized the AdaBoost algorithm to select the most relevant features for a particular classification application. We combined simple "base" analog classifiers to form a strong one. AdaBoost is widely used in the machine learning and digital signal processing communities, but we first adopted the algorithm to assist the design of an analog classifier.

    4. The AdaBoost-based analog detector was designed. We developed the circuits to realize the AdaBoost-based classifier design. The two-layer design consists of three single-input perceptrons in the first layer, and a weighted combination

of the first layer outputs are performed in the second stage. In one design, we used VMMs and current-mode comparators to implement both stages; the circuit was verified in SPICE. In another design, we used a simplified circuit to implement the classifier; the circuit is verified on the RASP 2.8a reconfigurable analog platform and can be generalized for larger classification systems.

9.2 Future Research

9.2.1 Sound Enhancement using System Identification

The sound enhancement algorithm described in Chapter 2 improves the sound quality of piezoelectric loudspeakers, but the enhanced audio signal still sounds weaker than those coming from a larger (electrodynamic) speaker. Part of the inferiority in sound quality is due to the physical size of the speaker, but nonlinearity also causes sound quality degradation. Lashkari introduced a method to reduce the nonlinearity by modeling the tiny (electrodynamic) speaker by a Volterra series and inverting (pre-distorting) the model [59]. This technique can be incorporated into our algorithm to further enhance the output sound quality.

9.2.2 Speaker Protection Algorithm in Feedback Configuration

The loudspeaker power management algorithm introduced in Chapter 3 is a feed-forward system. An incoming audio stream is analyzed and processed before being sent to the amplifier and speaker. We can improve the system's robustness by placing it in a feedback configuration, in which a measurement signal from the loudspeaker (e.g., resistance, which correlates with speaker temperature) can dynamically adjust the algorithm. For example, if the loudspeaker is overheated, the power reduction algorithm can be applied more aggressively. We can also include additional input signals to control the algorithm in real-time, tuning the hearing threshold according to the surrounding sound level.

9.2.3 Effects of Implementing Classifiers in the Analog Domain

Most classifiers are implemented in the digital domain, and the nonidealities of these systems are usually due to quantization effect. The adverse effects on classification accuracy

have been widely studied. For example, a statistical model was used to analyze the effects of quantization in neural networks [110]. The worst case analysis of weight inaccuracy due to quantization effect effects in perceptrons was also reported [5]. The analog classifiers presented in Chapters 6 and 8 exhibit different nonidealities, such as transistor mismatches, floating-gate transistor programming inaccuracy, and parasitic capacitance. Many of the analog nonidealities can be mitigated by pre-characterizing the devices on the FPAA. However, the effects on detection accuracy need to be carefully studied.

REFERENCES

[1] Aarts, R. M., "A comparison of some loudness measures for loudspeaker listening tests," *J. Audio Eng. Soc.*, vol. 40, no. 3, pp. 142–146, 1992.

[2] Abdalla, H. and Horiuchi, T., "Spike-based acoustic signal processing chips for detection and localization," in *Proc. IEEE Biomed. Circuits Syst. Conf.*, pp. 225–228, Nov. 2008.

[3] Alexandre, E., Cuadra, L., Rosa, M., and Lopez-Ferreras, F., "Feature selection for sound classification in hearing aids through restricted search driven by genetic algorithms," *IEEE Trans. Audio Speech Lang. Process.*, vol. 15, pp. 2249–2256, Nov. 2007.

[4] Anderson, D., *Audio signal enhancement using multi-resolution sinusoidal modeling*. PhD thesis, Georgia Institute of Technology, 1999.

[5] Anguita, D., Ridella, S., and Rovetta, S., "Worst case analysis of weight inaccuracy effects in multilayer perceptrons," *IEEE Trans. Neural Netw.*, vol. 10, pp. 415–418, Mar. 1999.

[6] Baker, M., Zhak, S., and Sarpeshkar, R., "A micropower envelope detector for audio applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'03)*, pp. V1–V4, May 2003.

[7] Baskaya, I. F., *Physical Design Automation for Large Scale Field Programmable Analog Arrays*. PhD thesis, Georgia Institute of Technology, 2009.

[8] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C., and Hasler, P., "A floating-gate-based field-programmable analog array," *IEEE J. Solid-State Circuits*, vol. 45, pp. 1781–1794, Sept. 2010.

[9] Basu, A., Twigg, C., Brink, S., Hasler, P., Petre, C., Ramakrishnan, S., Koziol, S., and Schlottmann, C., "RASP 2.8: A new generation of floating-gate based field programmable analog array," in *Proc. IEEE Custom Integr. Circuits Conf.*, pp. 213–216, Sept. 2008.

[10] Baxley, R. and Zhou, G., "Peak-to-Average Power Ratio Reduction," in *Wireless, Networking, Radar, Sensor Array Processing, and Nonlinear Signal Processing* (Madisetti, V., ed.), CRC, 2009.

[11] Baxter, K. C. and Fisher, K., "Gunshot detection sensor with display." United States Patent 8036065, Aug. 2011.

[12] Bishop, C. M., *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[13] Borgstrom, T., Ismail, M., and Bibyk, S., "Programmable current-mode neural network for implementation in analogue MOS VLSI," *IEE Proc. Circuits Devices Syst.*, vol. 137, pp. 175–178, Apr. 1990.

[14] Brandenburg, K., "MP3 and AAC explained," in *Audio Eng. Soc. Conf.: 17th Int. Conf.: High-Quality Audio Coding*, Aug. 1999.

[15] Burrow, S. and Grant, D., "Efficiency of low power audio amplifiers and loudspeakers," *IEEE Tran. Consum. Electron.*, vol. 47, pp. 622–630, Aug. 2001.

[16] Chabries, D., Anderson, D., Stockham, T.G., J., and Christiansen, R., "Application of a human auditory model to loudness perception and hearing compensation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'95)*, vol. 5, pp. 3527–3530, May 1995.

[17] Chakrabartty, S. and Cauwenberghs, G., "Margin normalization and propagation in analog VLSI," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, vol. 1, pp. I901–I904, May 2004.

[18] Chakrabartty, S. and Cauwenberghs, G., "Sub-microwatt analog VLSI trainable pattern classifier," *IEEE J. Solid-State Circuits*, vol. 42, pp. 1169–1179, May 2007.

[19] Chakrabartty, S. and Gore, A., "Sigma-delta analog to LPC feature converters for portable recognition interfaces," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'09)*, pp. 2673–2676, May 2009.

[20] Chan, V., Liu, S.-C., and van Schaik, A., "AER EAR: A matched silicon cochlea pair with address event representation interface," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, pp. 48–59, Jan. 2007.

[21] Chen, S., Cowan, C., and Grant, P., "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, pp. 302–309, Mar. 1991.

[22] Childers, D., Skinner, D., and Kemerait, R., "The cepstrum: A guide to processing," *Proc. IEEE*, vol. 65, pp. 1428–1443, Oct. 1977.

[23] Chiu, L. K., Gestner, B., and Anderson, D., "Design of analog audio classifiers with AdaBoost-Based feature selection," in *IEEE Int. Symp. Circuits Syst. (ISCAS'11)*, pp. 2469–2472, May 2011.

[24] Chu, W., *Auditory-Based Noise-Robust Audio Classification Algorithms*. PhD thesis, McGill University, 2008.

[25] Chu, W. and Champagne, B., "A noise-robust fft-based auditory spectrum with application in audio classification," *IEEE Trans. Audio Speech Lang. Process.*, vol. 16, pp. 137–150, Jan. 2008.

[26] Current, K., "Current-mode CMOS multiple-valued logic circuits," *IEEE J. Solid-State Circuits*, vol. 29, pp. 95–107, Feb. 1994.

[27] Davis, S. and Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 28, pp. 357–366, Aug. 1980.

[28] Delbrück, T., Koch, T., Berner, R., and Hermansky, H., "Fully integrated 500uw speech detection wake-up circuit," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'10)*, pp. 2015–2018, May 2010.

[29] Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern Classification*. New York, NY: Willey-Interscience, second ed., 2000.

[30] El-Maleh, K., Klein, M., Petrucci, G., and Kabal, P., "Speech/music discrimination for multimedia applications," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'00)*, vol. 4, pp. 2445–2448, June 2000.

[31] Elhilali, M., Chi, T., and Shamma, S. A., "A spectro-temporal modulation index (stmi) for assessment of speech intelligibility," *Speech Commun.*, vol. 41, pp. 331–348, Oct. 2003.

[32] Esmaili, S., Krishnan, S., and Raahemifar, K., "Content based audio classification and retrieval using joint time-frequency analysis," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'04)*, vol. 5, pp. V665–V668, May 2004.

[33] Freitas, D. and Current, K., "CMOS current comparator circuit," *IET Electronics Letters*, vol. 19, pp. 695–697, Aug. 1983.

[34] Furth, P. and Andreou, A., "Bit-energy comparison of discrete and continuous signal representations at the circuit level," in *4th Workshop on Physics and Computation*, Nov. 1997.

[35] Gatet, L., Tap-Beteille, H., and Lescure, M., "Analog neural network implementation for a real-time surface classification application," *IEEE Sensors J.*, vol. 8, pp. 1413 –1421, Aug. 2008.

[36] Gerstner, W. and Kistler, W., *Spiking Neuron Nodels: Single Neurons, Populations, Plasticity*. New York, NY: Cambridge Univ Press, 2002.

[37] Gestner, B., Tanner, J., and Anderson, D., "Glass break detector analog front-end using novel classifier circuit," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'07)*, pp. 3586–3589, May 2007.

[38] Graham, D., Hasler, P., Chawla, R., and Smith, P., "A low-power programmable bandpass filter section for higher order filter applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, pp. 1165–1176, June 2007.

[39] Gray, A., "What is machine learning? (overview)," lecture notes in Computational Data Analysis: Foundations of Machine Learning & Data Mining, Georgia Institute of Technology, 2009.

[40] Hasler, P., "Low-power programmable signal processing," in *Proc. IEEE Int. Workshop Syst.-on-Chip Real-Time Applicat.*, pp. 413–418, July 2005.

[41] Hasler, P. and Akers, L., "Implementation of analog neural networks," in *Proc. 10th Annu. Inter. Phoenix Conf. Comput. Commun.*, pp. 32–38, Mar. 1991.

[42] Hasler, P. and Akers, L., "Circuit implementation of trainable neural networks employing both supervised and unsupervised techniques," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'92)*, vol. 3, pp. 1565–1568, May 1992.

[43] Hasler, P. and Anderson, D., "Cooperative analog-digital signal processing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'02)*, vol. 4, pp. IV3972–IV3975, Aug. 2002.

[44] Hasler, P., Kozoil, S., Farquhar, E., and Basu, A., "Transistor channel dendrites implementing hmm classifiers," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'07)*, pp. 3359 –3362, May 2007.

[45] Hasler, P., Smith, P., Farquhar, E., and Anderson, D., "A neuromorphic ic connection between cortical dendritic processing and hmm classification," in *Proc. IEEE Workshop Digit. Signal Process.*, pp. 334–337, Aug. 2004.

[46] Hasler, P., Smith, P., Graham, D., Ellis, R., and Anderson, D., "Analog floating-gate, on-chip auditory sensing system interfaces," *IEEE Sensors J.*, vol. 5, pp. 1027–1034, Oct. 2005.

[47] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York, NY: Springer, 2009.

[48] Hayes, M. H., *Statistical Digital Signal Processing and Modeling.* New York, NY: John Wiley & Son, 1996.

[49] Horikawa, T. and Kobayashi, K., "Application of ceramic piezoelectric device to audio speaker," in *Proc. Soc. Instrum. Control Eng. Annu. Conf.*, pp. 1–4, Aug. 2008.

[50] Jaeger, H., "The "echo state" approach to analysing and training recurrent neural networks," Tech. Rep. GMD Report 148, German National Research Center for Information Technology, 2001.

[51] Jensen, S. and Solgaard, M. E., "Thermal protection of electro dynamic transducers used in loudspeaker systems." United States Patent 0257599 A1, Oct. 15 2009.

[52] Johnston, J., "Transform coding of audio signals using perceptual noise criteria," *IEEE J. Sel. Areas Commun.*, vol. 6, pp. 314–323, Feb. 1988.

[53] Kang, K. and Shibata, T., "An on-chip-trainable gaussian-kernel analog support vector machine," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, pp. 1513–1524, July 2010.

[54] Kedem, B., "Spectral analysis and discrimination by zero-crossings," *Proc. IEEE*, vol. 74, pp. 1477–1493, Nov. 1986.

[55] Koziol, S., Schlottmann, C., Basu, A., Brink, S., Petre, C., Degnan, B., Ramakrishnan, S., Hasler, P., and Balavoine, A., "Hardware and software infrastructure for a family of floating-gate based FPAAs," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'10)*, pp. 2794–2797, June 2010.

[56] Kumar, N., Cauwenberghs, G., and Andreou, A., "Level crossing time interval circuit for micro-power analog VLSI auditory processing," in *Proc. IEEE Workshop Neural Netw. Signal Process.*, pp. 581–590, Aug. 1995.

[57] Kumar, N., Himmelbauer, W., Cauwenberghs, G., and Andreou, A., "An analog VLSI chip with asynchronous interface for auditory feature extraction," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, pp. 600–606, May 1998.

[58] Larsen, E. and Aarts, R., *Audio Bandwidth Extension: Application of Psychoacoustics, Signal Processing, and Loudspeaker Design.* Wiley Chichester, 2004.

[59] Lashkari, K., "High quality sound from small loudspeakers using the exact inverse," in *Conf. Rec. 38th Asilomar Conf. Signal Syst. Comput.*, vol. 1, pp. 430–434, Nov. 2004.

[60] Li, S., "Content-based audio classification and retrieval using the nearest feature line method," *IEEE Trans. Speech Audio Process.*, vol. 8, pp. 619–625, Sept. 2000.

[61] Lin, C.-C., Chen, S.-H., Truong, T.-K., and Chang, Y., "Audio classification and categorization based on wavelets and support vector machine," *IEEE Trans. Speech Audio Process.*, vol. 13, pp. 644–651, Sept. 2005.

[62] Liu, S.-C., Kramer, J., Indiveri, G., Delbrück, T., and Douglas, R., *Analog VLSI: Circuits and Principles.* Cambridge, MA: MIT Press, 2002.

[63] Liu, Z., Huang, J., Wang, Y., and Chen, T., "Audio feature extraction and analysis for scene classification," in *Proc. IEEE 1st Workshop Multimedia Signal Process.*, pp. 343–348, June 1997.

[64] Lont, J. and Guggenbühl, W., "Analog CMOS implementation of a multilayer perceptron with nonlinear synapses," *IEEE Trans. Neural Netw.*, vol. 3, pp. 457–465, May 1992.

[65] Lu, L., Zhang, H.-J., and Jiang, H., "Content analysis for audio classification and segmentation," *IEEE Trans. Speech Audio Process.*, vol. 10, pp. 504–516, Oct. 2002.

[66] Lyon, R. and Mead, C., "An analog electronic cochlea," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 36, pp. 1119 –1134, July 1988.

[67] Maass, W., Natschläger, T., and Markram, H., "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002.

[68] Mead, C., *Analog VLSI and Neutral Systems.* Addison-Wesley, 1989.

[69] Milev, M. and Hristov, M., "Analog implementation of ann with inherent quadratic nonlinearity of the synapses," *IEEE Trans. Neural Netw.*, vol. 14, pp. 1187–1200, Sept. 2003.

[70] Montalvo, A., Gyurcsik, R., and Paulos, J., "Toward a general-purpose analog VLSI neural network with on-chip learning," *IEEE Trans. Neural Netw.*, vol. 8, pp. 413–423, Mar. 1997.

[71] Moore, B., *An Introduction to the Psychology of Hearing.* Academic Press, 2003.

[72] Neunaber, B., "Apparatus and method of limiting power applied to a loudspeaker," Sept. 2005.

[73] Odame, K., Anderson, D., and Hasler, P., "A bandpass filter with inherent gain adaptation for hearing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, pp. 786–795, Apr. 2008.

[74] Ohga, J., Takei, T., and Moriyama, N., "Wideband piezoelectric rectangular loudspeakers using a tuck shaped pvdf bimorph," *IEEE Trans. Dielectr. Electr. Insul.*, vol. 17, pp. 1074–1078, Aug. 2010.

[75] Oppenheim, A., Schafer, R., and Stockham, T., J., "Nonlinear filtering of multiplied and convolved signals," *IEEE Trans. Audio Electroacoust.*, vol. 16, pp. 437–466, Sept. 1968.

[76] Oster, M., Wang, Y., Douglas, R., and Liu, S.-C., "Quantification of a spike-based winner-take-all VLSI network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, pp. 3160–3169, Nov. 2008.

[77] Painter, T. and Spanias, A., "Perceptual coding of digital audio," *Proc. IEEE*, vol. 88, pp. 451–515, Apr. 2000.

[78] Papoulis, A. and Pillai, S. U., *Probability, Random Variables and Stochastic Processes.* New York, NY: McGraw-Hill, 2002.

[79] Peng, S.-Y., Hasler, P., and Anderson, D., "An analog programmable multidimensional radial basis function based classifier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, pp. 2148–2158, Oct. 2007.

[80] Peng, S.-Y., Minch, B., and Hasler, P., "A programmable floating-gate bump circuit with variable width," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05)*, vol. 5, pp. 4341–4344, May 2005.

[81] Punzenberger, M. and Enz, C., "A 1.2-V low-power BiCMOS class AB log-domain filter," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1968–1978, Dec. 1997.

[82] Quatieri, T., *Discrete-Time Speech Signal Processing.* Upper Saddle River, NJ: Prentice Hall PTR, 2002.

[83] Quatieri, T. and McAulay, R., "Peak-to-rms reduction of speech based on a sinusoidal model," *IEEE Tran. Signal Process.*, vol. 39, pp. 273–288, Feb. 1991.

[84] Ramakrishnan, S., Basu, A., Chiu, L. K., Hasler, P., Anderson, D., and Brink, S., "Reconfigurable platform for implementing speech processing algorithms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, submitted for publication.

[85] Ravindran, S., *Physiologically Motivated Methods for Audio Pattern Classification.* PhD thesis, Georgia Institute of Technology, 2006.

[86] Ravindran, S. and Anderson, D., "Boosting as a dimensionality reduction tool for audio classification," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, vol. 3, pp. 465–468, 2004.

[87] Ravindran, S., Schlemmer, K., and Anderson, D., "A physiologically inspired method for audio classification," *EURASIP J. Appl. Signal Process.*, vol. 9, pp. 1374–1381, 2005.

[88] Re, E. F., Schaik, A. V., and Vittoz, E., "Linear predictive coding of speech using an analogue cochlear model," in *Proc. EuroSpeech*, vol. 1, pp. 119–122, 1995.

[89] Sarpeshkar, R., "Analog versus digital: extrapolating from electronics to neurobiology," *Neural Comput.*, vol. 10, pp. 1601–1638, Oct. 1998.

[90] Sarpeshkar, R., Lyon, R. F., and Mead, C., "A low-power wide-linear-range transconductance amplifier," *Analog Integrated Circuits and Signal Processing*, vol. 13, no. 1-2, pp. 123–151, 1997.

[91] Sarpeshkar, R., Lyon, R. F., and Mead, C., "A low-power wide-dynamic-range analog VLSI cochlea," *Analog Integrated Circuits and Signal Processing*, pp. 49–103, 1998.

[92] Scheirer, E. and Slaney, M., "Construction and evaluation of a robust multifeature speech/music discriminator," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'97)*, vol. 2, pp. 1331–1334, Apr. 1997.

[93] Schlottmann, C. and Hasler, P., "A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 1, pp. 403–411, Sept. 2011.

[94] Schlottmann, C., Petre, C., and Hasler, P., "A high-level simulink-based tool for FPAA configuration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, pp. 10–18, Jan. 2012.

[95] Schmidt, J. and Rutledge, J., "Multichannel dynamic range compression for music signals," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process (ICASSP'96)*, vol. 2, pp. 1013–1016, May 1996.

[96] Sedra, A., Roberts, G., and Gohh, F., "The current conveyor: history, progress and new results," *IEE Proc. Circuits Devices Syst.*, vol. 137, pp. 78–87, Apr. 1990.

[97] Serrano, G. and Hasler, P., "A floating-gate dac array," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, vol. 1, pp. I–357–I–360, May 2004.

[98] Smith, P., *An Analog Architecture for Auditory Feature Extraction and Recognition.* PhD thesis, Georgia Institute of Technology, 2004.

[99] Smith, P., Graham, D., Chawla, R., and Hasler, P., "A five-transistor bandpass filter element," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, vol. 1, pp. I861–I864, May 2004.

[100] Smith, P. and Hasler, P., "Analog speech recognition project," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'02)*, vol. 4, pp. IV3988–IV3991, May 2002.

[101] Tieu, K. and Viola, P., "Boosting image retrieval," *Int. J. Comput. Vision*, pp. 228–235, 2000.

[102] Twigg, C., *Floating Gate Based Large-Scale Field-Programmable Analog Arrays for Analog Signal Processing*. PhD thesis, Georgia Institute of Technology, 2006.

[103] Twigg, C., Gray, J., and Hasler, P., "Programmable floating gate FPAA switches are not dead weight," in *IEEE Int. Symp. Circuits Syst. (ISCAS'07)*, pp. 169–172, May 2007.

[104] van Schaik, A., Fragnière, E., , and Vittoz, E., "Improved silicon cochlea using compatible lateral bipolar transistors," in *Advances in Neural Information Processing Systems 8*, pp. 671–677, MIT Press, Nov. 1995.

[105] Visser, E., Toman, J., and Chan, K., "Robust separation of speech signals in a noisy environment." United States Patent 7464029, Dec. 2008.

[106] Wang, K. and Shamma, S., "Self-normalization and noise-robustness in early auditory representations," *IEEE Trans. Speech Audio Process.*, vol. 2, pp. 421–435, July 1994.

[107] Wasserman, L., *All of Statistics*. New York, NY: Springer, 2004.

[108] Watts, L., Kerns, D., Lyon, R., and Mead, C., "Improved implementation of the silicon cochlea," *IEEE J. Solid-State Circuits*, vol. 27, pp. 692–700, May 1992.

[109] Wold, E., Blum, T., Keislar, D., and Wheaten, J., "Content-based classification, search, and retrieval of audio," *IEEE Multimedia*, vol. 3, pp. 27–36, Fall 1996.

[110] Xie, Y. and Jabri, M., "Analysis of the effects of quantization in multilayer neural networks using a statistical model," *IEEE Trans. Neural Netw.*, vol. 3, pp. 334–338, Mar. 1992.

[111] Yang, X., Wang, K., and Shamma, S., "Auditory representations of acoustic signals," *IEEE Trans. Information Theory*, vol. 38, pp. 824–839, mar 1992.

[112] Zatorre, G., Medrano, N., Sanz, M., Calvo, B., Martinez, P., and Celma, S., "Designing adaptive conditioning electronics for smart sensing," *IEEE Sensors J.*, vol. 10, pp. 831–838, Apr. 2010.

[113] Zhak, S., Baker, M., and Sarpeshkar, R., "A low-power wide dynamic range envelope detector," *IEEE J. Solid-State Circuits*, vol. 38, pp. 1750–1753, Oct. 2003.

[114] Zhang, T. and Jay Kuo, C.-C., "Audio content analysis for online audiovisual data segmentation and classification," *IEEE Trans. Speech Audio Process.*, vol. 9, pp. 441–457, May 2001.